Avoka Transact Fluent SDK

Avoka Transact is designed to be integrated with other systems in many different ways. This makes the platform highly extensible which helps meet the demands of the most complex financial and government environments. Transact 5 introduces a new Fluent API and desktop IDE support, which make the development of extension points even easier than before.

Transact allows the demands of complex financial and government environments to be met with low or no code – using configuration rather than development to achieve the majority of requirements. However, sometimes customers need to extend the core capabilities of Transact to achieve specific goals not possible as a built-in capability – for example, integration to a custom back-end system. Transact is designed to be open and extensible, so that these types of extensions can be seamlessly integrated into the core Transact system. Transact includes over twenty different extension points – points in the lifecycle of a transaction – where developers can “plug in” code to extend the transaction platform to create a uniquely tailored transaction process.

The Fluent SDK provides IDE-based development, including source control integration for team development projects, code modularization and reuse, unit testing, and real-time code completion and error checking. The Fluent API has been designed from the ground up to provide a simplified Fluent programming style. This API and programming is designed for improved learning and reduced complexity, higher productivity, and increased code readability. The Groovy language which our developers find so productive is still used. Upgrades and maintenance become smoother because of the simplified and unchanging Fluent API. And finally, the new API provides enhanced levels of security and performance. The older Groovy API is still available for backward compatibility.

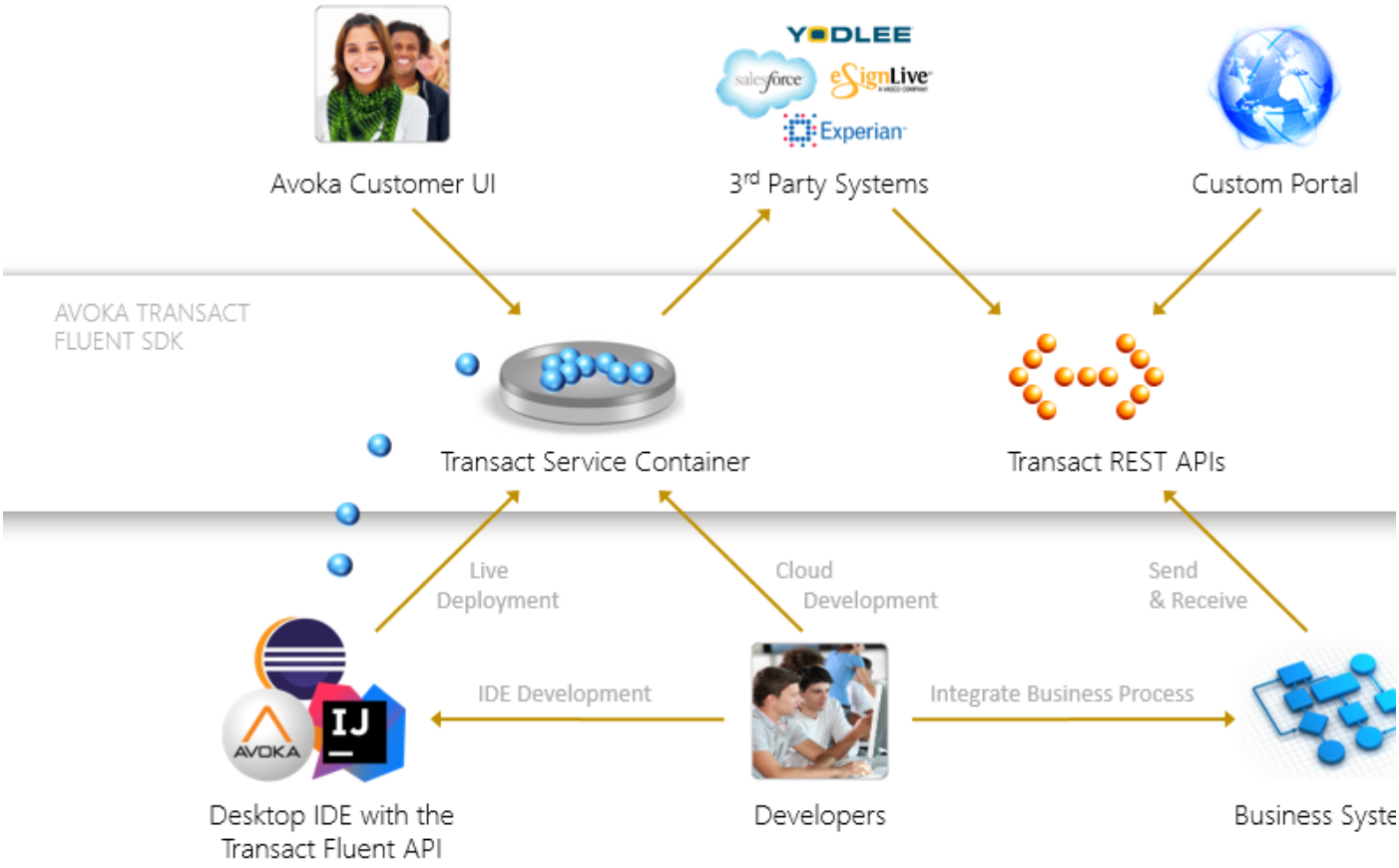
The new SDK and API create an environment for customers, partners and the Avoka service team to rapidly extend Transact with new back office integrations, new workflows, and new connectors to FinTech services. No other customer acquisition platform offers the ease and range of extensibility as Transact 5, designed to meet the needs of the world’s largest financial and government institutions.

The Avoka Transact Fluent SDK consists of the following elements:

1. The Transact Fluent API
   A set of Java libraries for engineers to develop new services against, including capabilities to interface with the Avoka Transact platform and a bunch of commonly used 3rd Party Java Libraries. Groovy services produced using the Transact Fluent API can run in the **Transact Service Container** to enhance or augment customer engagements and interface with external party systems. Services can be developed in a modern desktop IDE (recommended) or in the cloud (using the web interface to Transact Manager).

2. The Transact Project Template
   A pre-configured project structure containing Ant build files and features used when developing, deploying and testing Transact extensions using a modern desktop IDE.

3. Transact REST APIs
   A published set of REST APIs designed to support inbound integrations from 3rd party systems, core business systems and custom portals. These APIs provide an interface for external systems to perform key actions such as retrieval of transaction lists, delivery of completed submission, and assignment of form based tasks to anonymous and/or registered users.



This documentation space contains information to get you up and running for development against the Transact Fluent SDK and is structured as follows:

- Fluent Groovy Service Developer Guide - A comprehensive guide for Fluent Groovy Service development with the Transact Fluent SDK
- Transact Service Types - A definitive list of the service types available to you, serving as extension points for the platform
- Transact Fluent API Reference - Javadoc style reference for the Transact Fluent API
- Transact REST API Reference - Reference documentation for the published REST APIs provided in the Transact Fluent SDK
- Transact Fluent SDK Download - Assets required for desktop IDE development of Groovy services

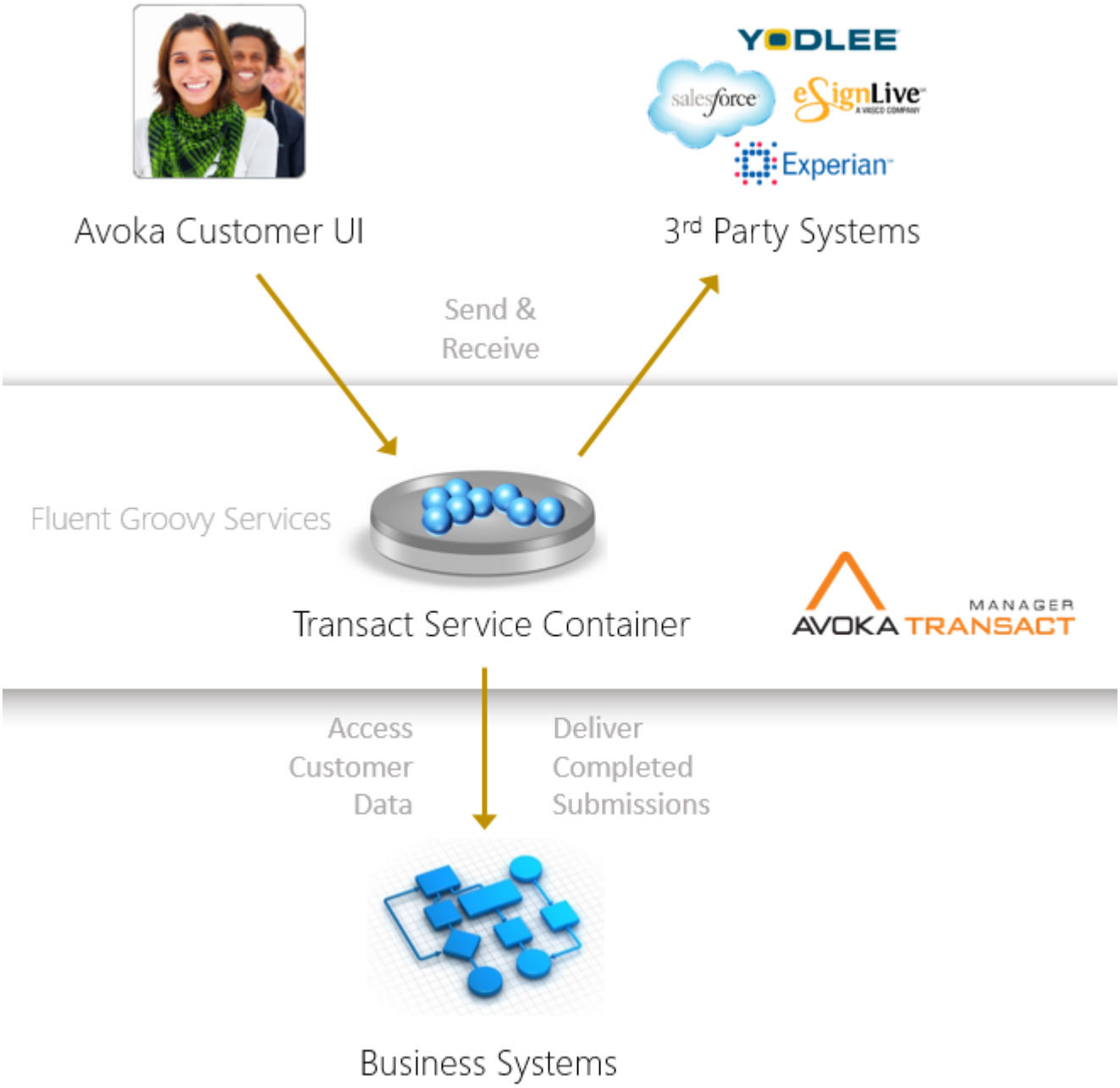
Fluent Groovy Service Developer Guide

The Fluent Groovy Service Developer Guide is designed for Avoka customers, partners and staff who need to extend the capabilities of Avoka Transact to access remote systems or perform some server-side logic to effect the life cycle of a transaction.

> Developing Fluent Groovy Services is a coding exercise and like any coding, requires technical skills and an understanding of software development principles.

There are many reasons why you would want to develop a Fluent Groovy Service to run in the Transact Service Container. Some examples are:

- Retrieve information about an existing customer to pre-fill a form when they open it
- Validate information entered into a form by a user against a remote API
- Identify an individual by accessing remote identity services
- Allow a form user to search and retrieve information from a remote system
- Provide real-time decisioning into a product application form by calling a remote decisioning system and assessing the response
- Update a remote system with information entered by the user as they move between pages of a form
- Deliver completed form submissions to down stream systems



Getting Started with Fluent Groovy Services

The Groovy programming languiage is built on Java and the grammar of the language derives from the Java grammar, but enhances it with specific constructs and allows certain simplifications that improve developer productivity while making code more concise and readable. Given the foundation on Java, most of the Java Syntax is supported in Groovy and developers with strong Java experience will pickup Groovy faster than those without.

To get started with Fluent Groovy service development, we recommend you follow the following sequence of steps:

1. Complete your Development Environment Setup in the IDE of your choice.
2. Follow the Hello World Service Example to develop, deploy and test your first service.
3. Review the Transact Fluent API Reference documentation to build your awareness of the packages and classes it contains.
4. Skim through the article on Working with the Ant Build Files to understand the build functions included in The Transact Project Template.
5. Review our articles on Coding Conventions & Practices and Techniques for Solving Common Requirements.

For developers new to Groovy, we strongly recommend you review the following assets:

- Read through our introductory content relating to Groovy Declarations and Control Statements.
- Official - It is highly recommended that anyone starting off with Groovy review the Official Groovy Documentation in depth.
- This 1 hour Groovy Tutoral on YouTube covers a great deal of content very quickly and is recommended viewing.

Related Technologies

Groovy services developed on the Avoka platform are commonly accessed via a RESTful web service API so an understanding of this protocol will also be beneficial. IDE support is facilitated by Ant build scripts and knowledge of Ant will be required to customize the build function.

The following references should be reviewed by developers involved in groovy service development on the Avoka Transact platform:

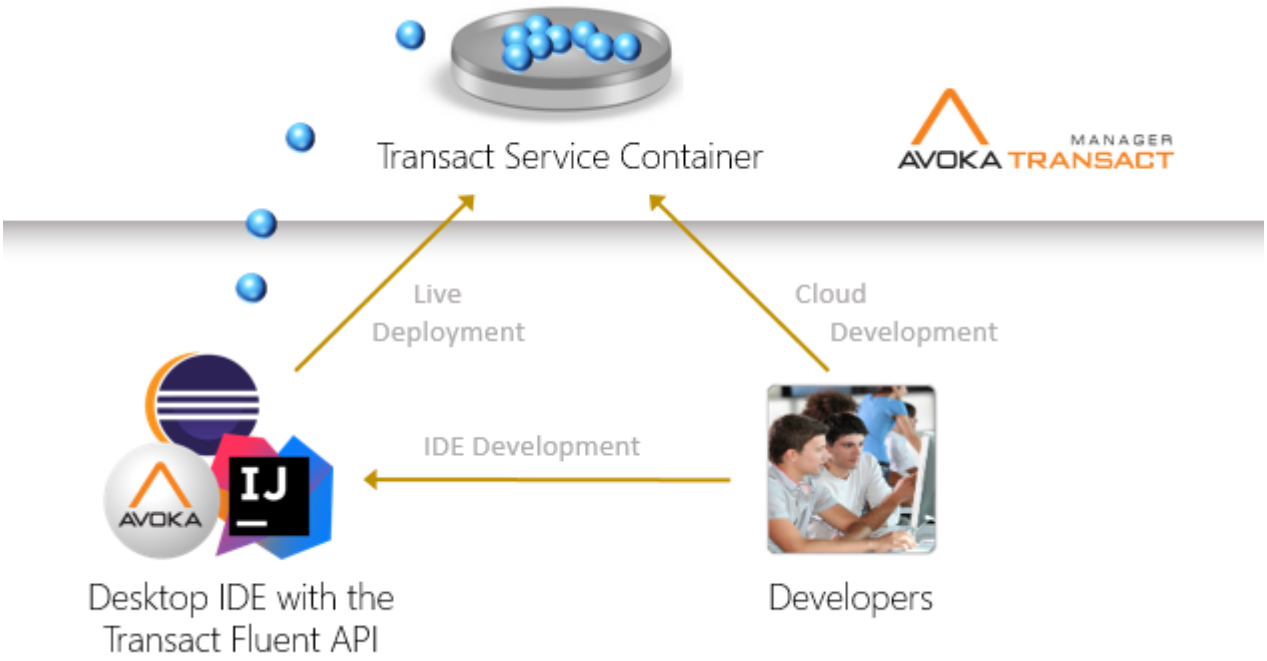
- RESTful Web services - A nice concise description of RESTful Web services by IBM.
- Apache Ant - Ant is used to facilitate packaging, uploading and testing of Groovy services from IDEs.
- JSON Home & JSON Tutorial - The JSON format is commonly used for managing structured data in Groovy and REST.

Development Environment Setup

Avoka Transact supports 2 different development environments which may be used in isolation or together depending on your need and your preference.

1. Desktop IDE Development with The Transact Fluent API
   Development is done in the desktop IDE of your preference by configuring the Transact Fluent API. This is the recommended approach for service development as it supports a superior developer experience with code-complete, source control integration and additional configuration options.

2. Cloud-Based Development in Transact Manager
   All development is done directly in the Transact Service Container on the Transact Manager server, hosted in the cloud or managed infrastructure. This approach requires no setup but lacks the benefits and convenience offered by modern IDEs.



To build and execute Fluent Groovy Services for Avoka Transact you must have access to a Transact Manager development instance. It is recommended that you create your own organization in Manager to isolate your work from others.

Desktop IDE Development with The Transact Fluent API

Avoka recommends that all Groovy service development be done in a modern Desktop IDE supporting Java and Groovy programming languages.

Configuring your IDE involves downloading and configuring the Transact Fluent SDK.

- Transact Fluent SDK Download

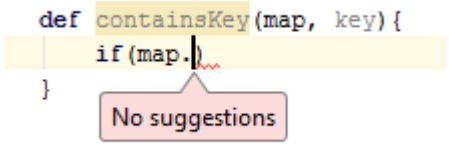
We have provided some specific instructions for configuring the most popular IDEs for Groovy service development:

- Configuring the SDK in IntelliJ
- Configuring the SDK in Eclipse

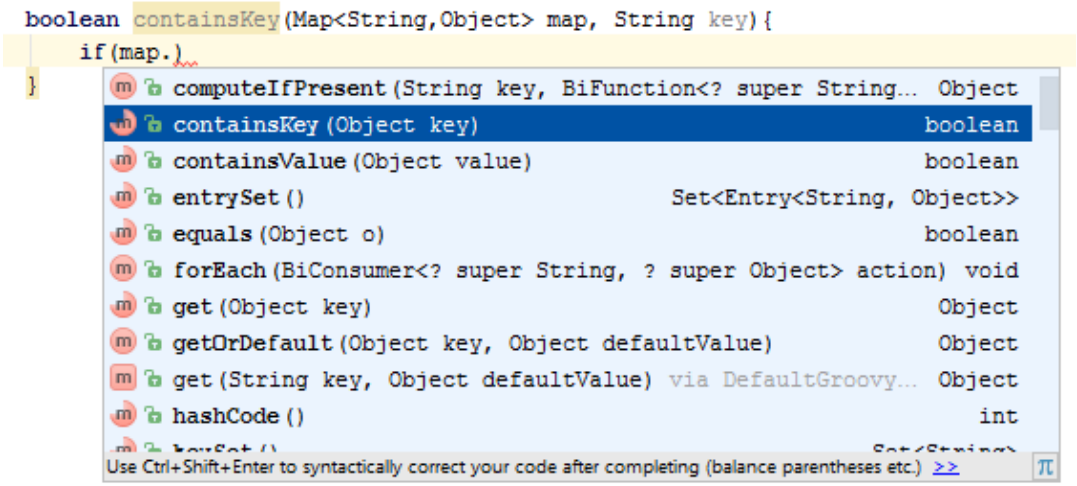
Why use an IDE?

Modern desktop IDEs such as Eclipse and IntelliJ are very popular because they provide an exceptional experience for the software developer, allowing them to achieve levels of productivity well above what they may be able to achieve otherwise.Support for IDE based development of Groovy services for the Transact platform gives developers access to features like real-time syntax and error checking, automatic code completion, code modularization for reuse, advanced refactoring functions, and source control repositories,

Perhaps the biggest productivity gain from using IDEs for Groovy development are provided by the code completion functions. IDE's are able to provide better code complete functions and type checking where defined types are used, making development and maintenance of Groovy code far more productive. In the method declaration below we're expecting the first parameter to be a Map but the IDE does not know that and cannot provide any suggestions as to what functions and attributes are available on that object:



Using defined types activates the IDE productivity assistants:



Configuring the SDK in IntelliJ

Watch the video

and/or follow the documented process below.

Before commencing this configuration please ensure you have installed IntelliJ (note the free community edition of IntelliJ is sufficient) and downloaded the Transact SDK assets:

- Transact Fluent SDK Download
- IntelliJ Download

If you already have a recent version of IntelliJ installed, you may wish to create a new workspace for development of Transact services.

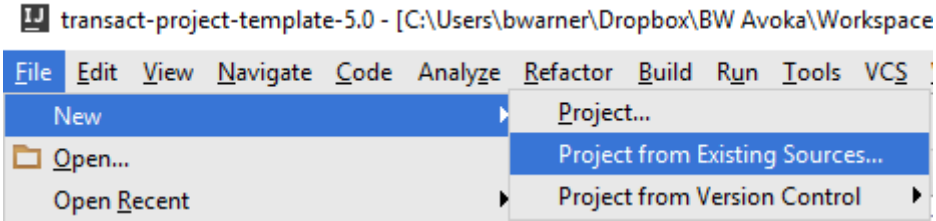
Dependencies

Ensure you have the correct version of Java installed and configured in your IDE as listed in the Dependencies section on the Fluent SDK Download page.

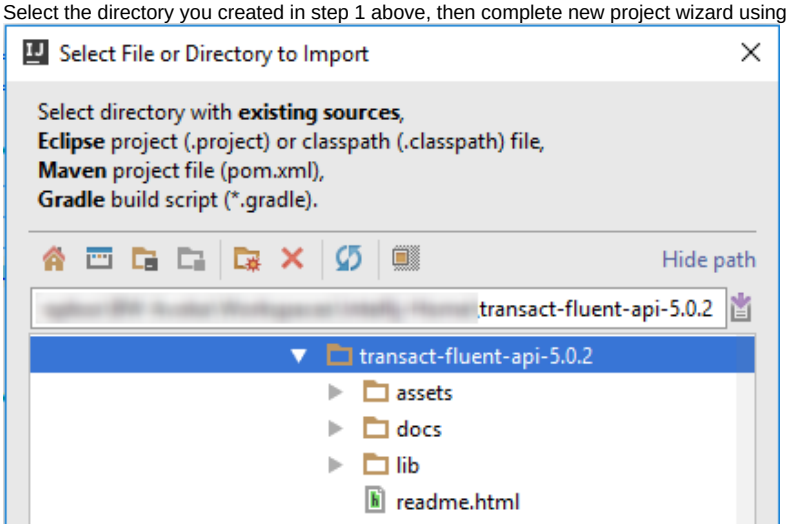
Create the transact-fluent-api-x.x.x Global Library

1. Unzip the **transact-fluent-api-x.x.x** file as a sub-folder of the same name in your IntelliJ workspace directory.
2. In IntelliJ, if opening for the first time select Import Project... from the spash screen.

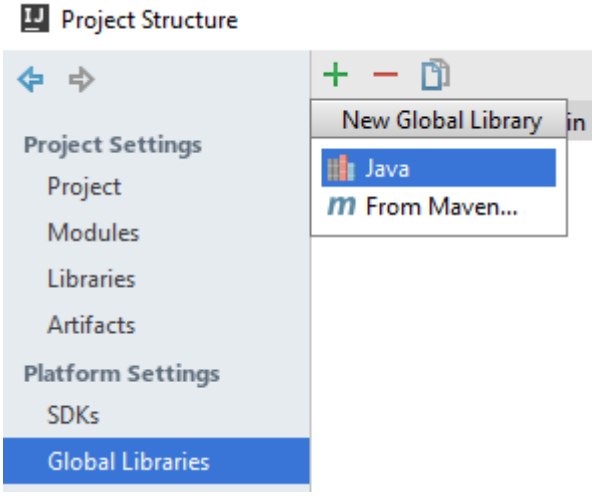
Otherwise, from the **File** menu in the editor select **New** > **Project from Existing Sources...**



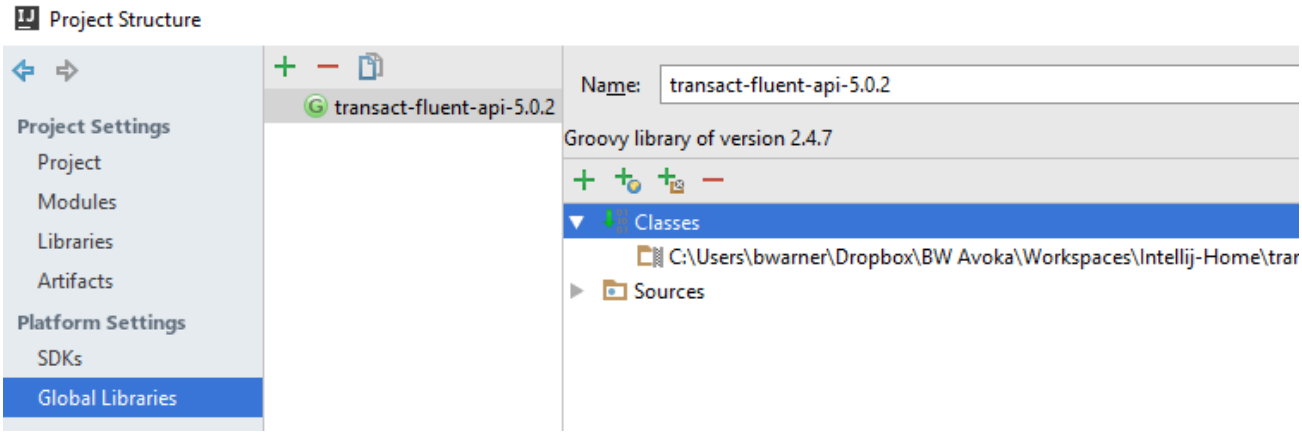
3. Select the directory you created in step 1 above, then complete new project wizard using defaults.



4. In the **Project Structure** dialog under **Global Libraries** click the **plus** sign to add a **Java** Library.



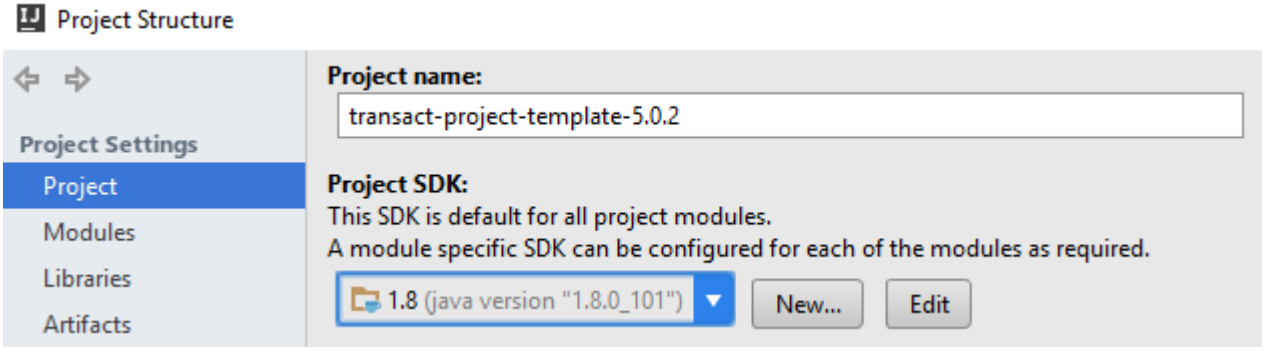
5. Select the **lib** directory inside the current project and click **OK.**
6. Rename the library to **transact-fluent-api-x.x.x** (match the project name) and Close.



Configure the Transact Project Template

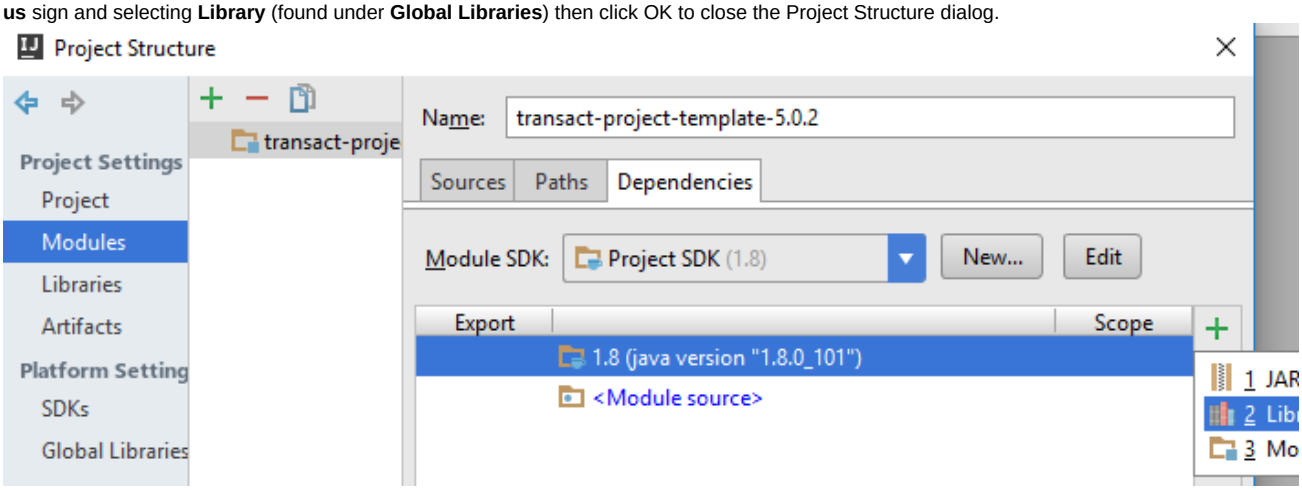
1. Unzip the **transact-project-template-x.x.x** file as a sub-folder of the same name in your IntelliJ workspace directory (use the same parent folder as used above so that it sits along side the **transact-fluent-api-x.x.x** folder).
2. In IntelliJ, from the **File** menu select **Open**, then locate the directory you created in step 1 (it should be recognized by IntelliJ as a project)

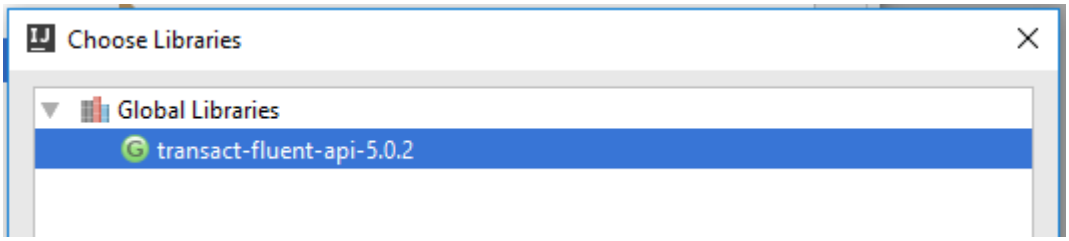
3. In the Project Structure dialog under Projects, ensure that Java 1.8 is selected as the Project SDK



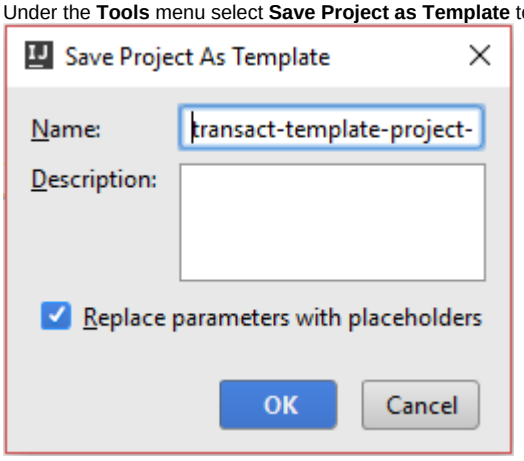
If you have not configured your Java SDK already, you will need to do this now by clicking the **New...** button.

4. In the **Project Structure** dialog under **Modules** add the **transact-fluent-api-x.x.x** global library configured earlier as a dependency using the **plus** sign and selecting **Library** (found under **Global Libraries**) then click OK to close the Project Structure dialog.





5. Open the **build.properties** file and ensure the **transact.api.dir** property exists and points to the **transact-fluent-api-x.x.x** root directory (use either relative or absolute path - default. **../transact-fluent-api-x.x.x**)
   Note if your template project shares a parent directory with the transact-fluent-api then the default value in this file should be fine.
6. Under the **Tools** menu select **Save Project as Template** to create a project template based on the current project, accept defaults and **Save**.

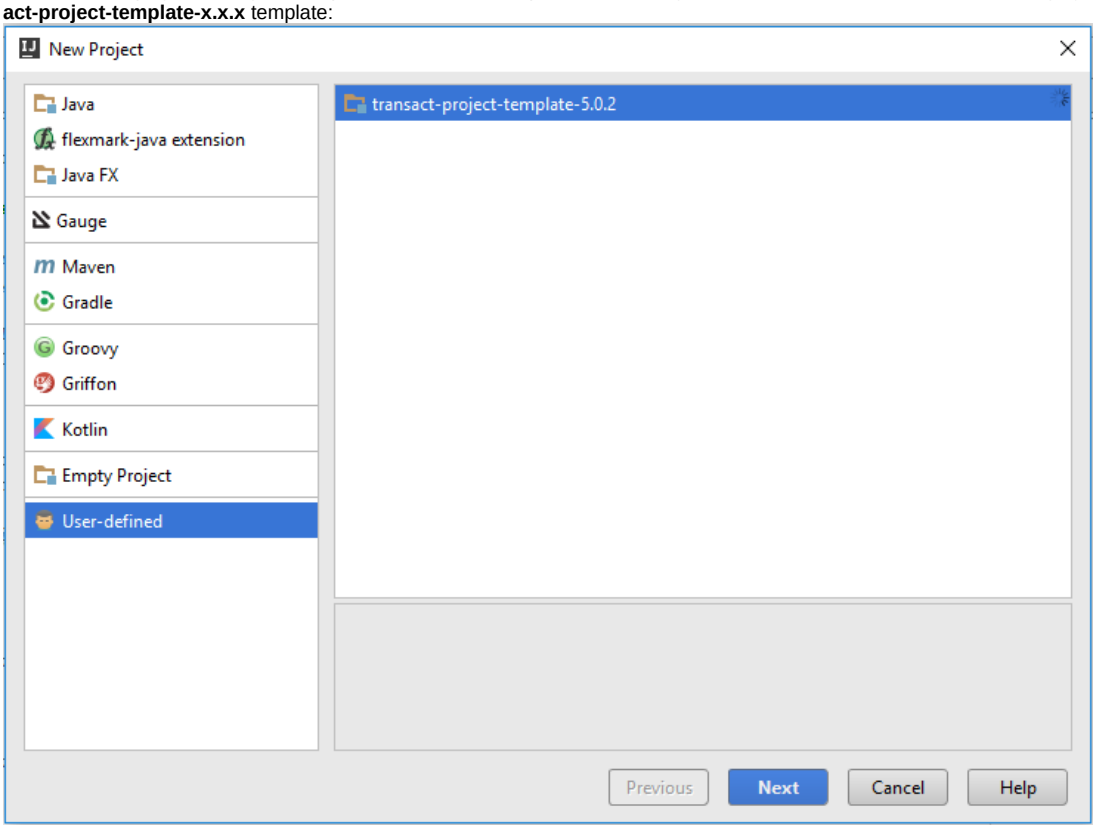


Create a new Transact Project

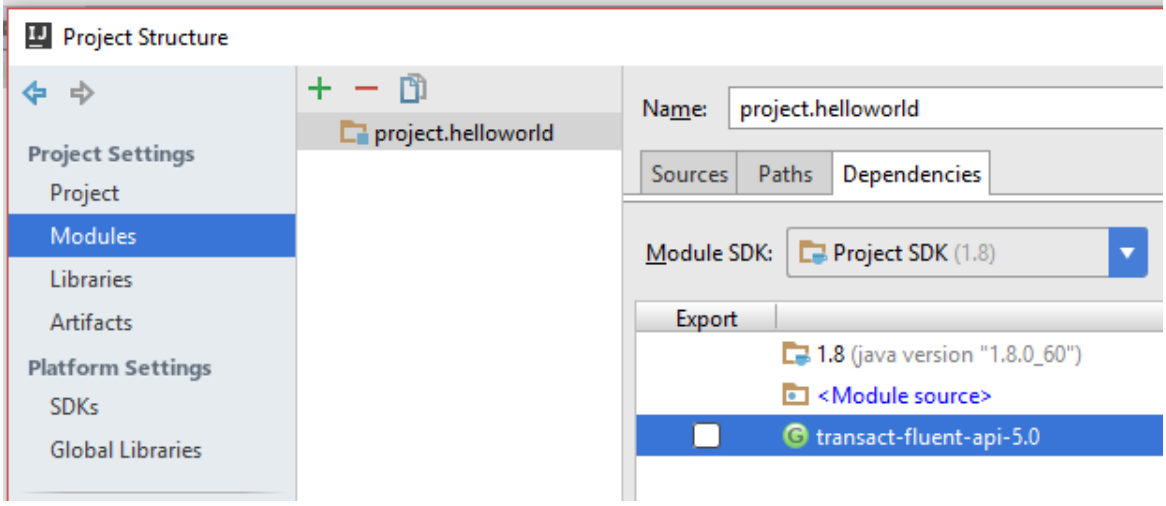
You may want to try this out by following the Hello World Service Example

When starting a new development project you'll want to setup a new IntelliJ project using the steps below:

1. To create new projects using the template created above, open the New Project Wizard, select the **User-defined** category and chose the **transact-project-template-x.x.x** template:



2. Give the project a name (e.g. **project.helloworld**) and click **Finish.**
3. Ensure that the **transact-fluent-api-x.x.x** Library is available to the project module (as you did when creating the project template above).



4. Open your **build.properties** file in the editor and modify the **project.name** and **project.code** properties as documented.
5. Open your **scaffold.xml** file in the **Ant Build** tab and run the **scaffold** target to initialize the project for your version of the SDK.

Generating new Groovy Services

When ever you are creating new services in the project, follow this procedure:

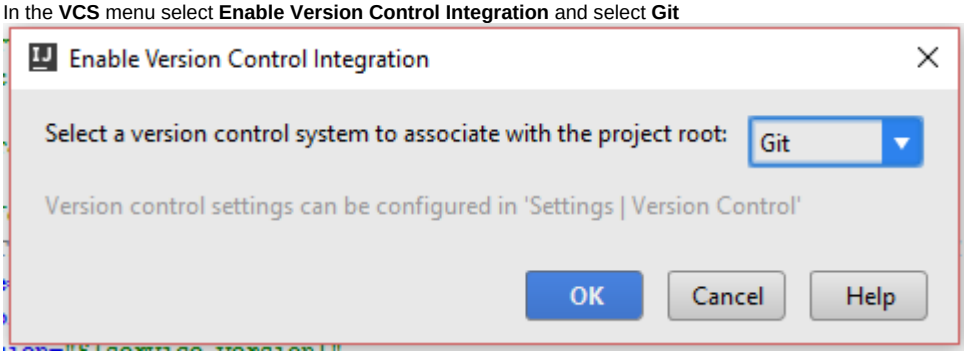
1. Open your **new-service.properties** file in the editor and modify the service generation properties as documented.
2. Open the **build.xml** file in the **Ant Build** tab and run the **create-new-service** build target.
3. Confirm that a new **build-svc-<service.code>.xml** file was created in the project root directory and that a new sub-folder was created in the **src** directory containing the new service files.
4. Open your **build-svc-<service-code>.xml** file in the **Ant Build** tab.

You are now ready to develop and deploy your new service.

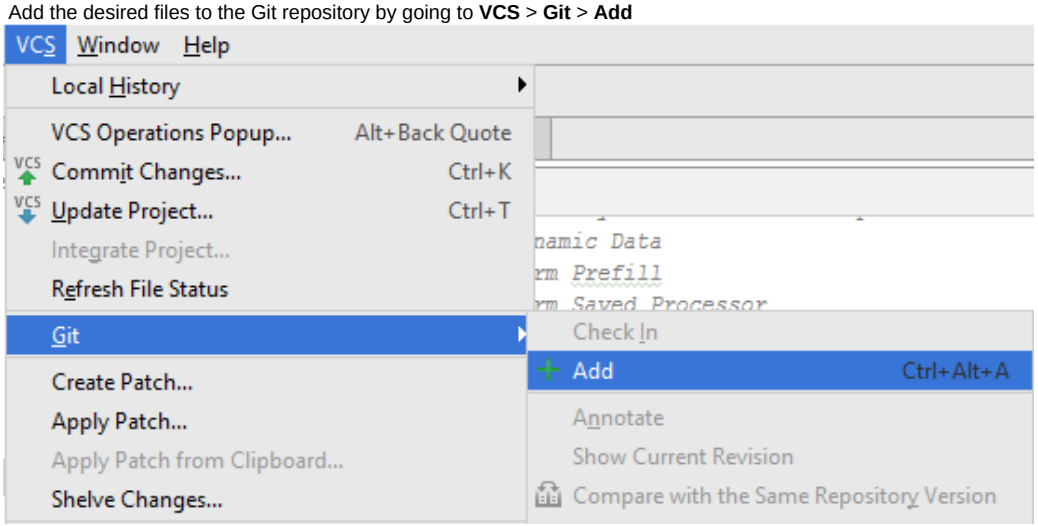
Linking your new Project to a Git Repository

Once you've created your new project you may wish to link it to a Git repository. This can be done as follows:

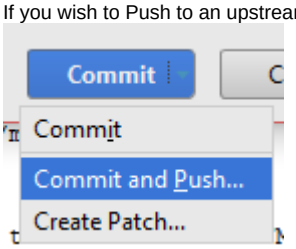
1. In the **VCS** menu select **Enable Version Control Integration** and select **Git**



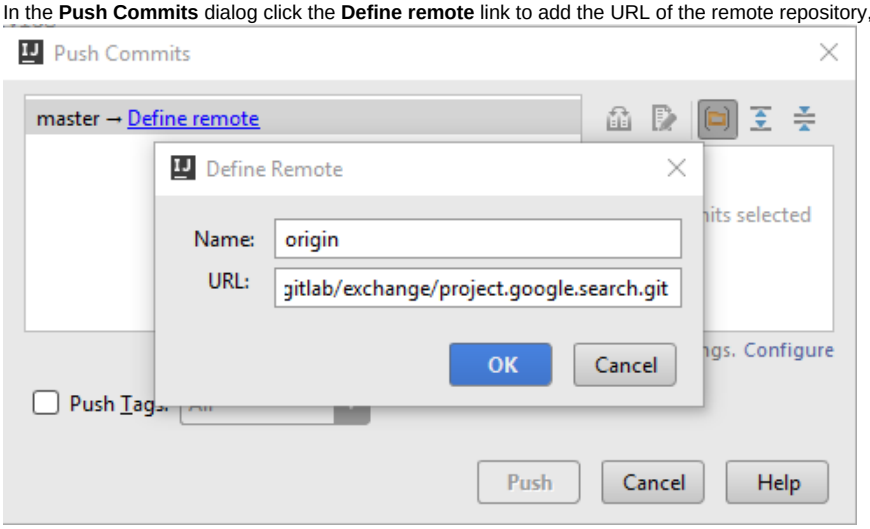
2. Add the desired files to the Git repository by going to **VCS** > **Git** > **Add**



The files you add should appear green in the Project structure.
Note: the following IDE generated files should be stored with the Git project (found in the project root directory or the .idea folder):
   a. **<project-name>.iml**
   b. **modules.xml**
   c. **ant.xml**
   d. **misc.xml**
3. You can now Commit these files to the Git repository in the standard manner.
4. If you wish to Push to an upstream repository, when committing select **Commit and Push**



5. In the **Push Commits** dialog click the **Define remote** link to add the URL of the remote repository, then click Push:



Configuring the SDK in Eclipse

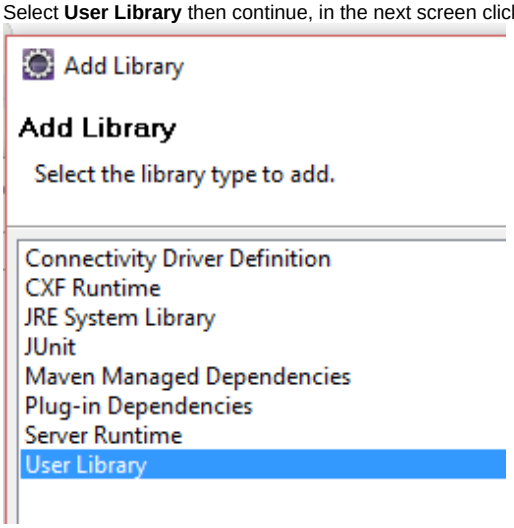
Before commencing this configuration please ensure you have installed Eclipse and downloaded the Transact SDK assets:

- Transact Fluent SDK Download
- Eclipse Download

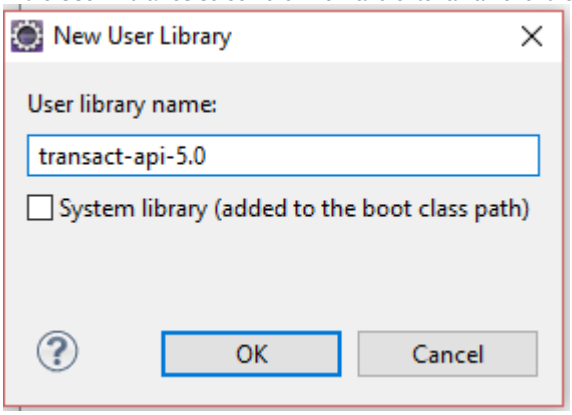
If you already have a recent version of Eclipse installed, you may wish to create a new workspace for development of Transact services.

Create the transact-fluent-api Global User Library

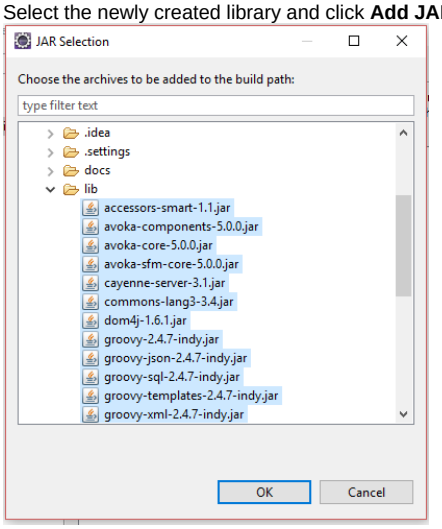
1. Unzip the **transact-fluent-api-x.x** file as a sub-folder of the same name in your Eclipse workspace directory.
2. Open the new directory as a project in Eclipse.
3. In the **Project Properties** dialog under **Java Build Path** select the **Libraries** tab and click **Add Library** to start the Add Library wizard
4. Select **User Library** then continue, in the next screen click the User Libraries button so we can add the library



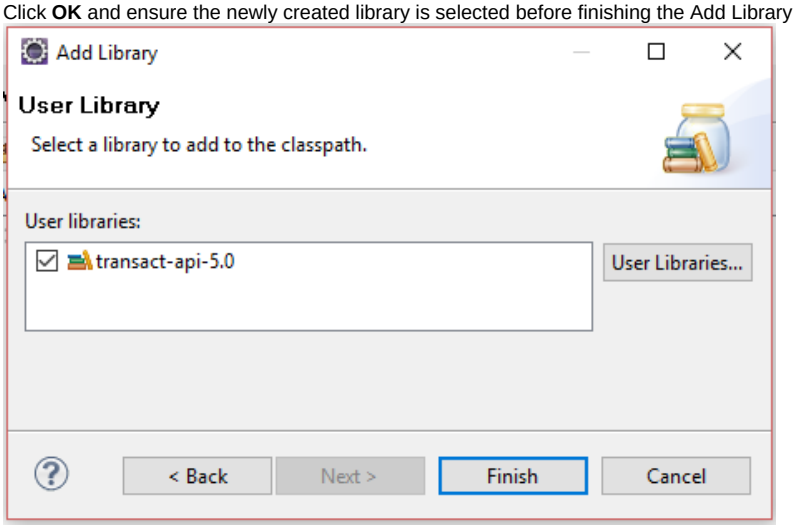
5. In the **User Libraries** screen click **New** and enter a name for the new library (e.g. **transact-fluent-api-x.x**), then click **OK**



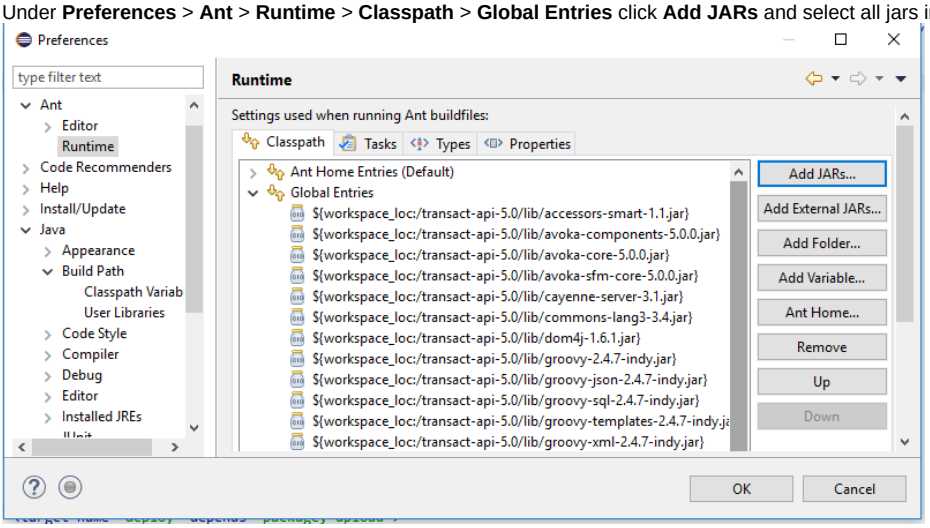
6. Select the newly created library and click **Add JARs** then select all jar files in the **lib** directory of the project to add



7. Click **OK** and ensure the newly created library is selected before finishing the Add Library wizard



8. You can also add these jar files to the Ant runtime global classpath if you wish, to save having to add these to the classpath of each build file. Under **Preferences** > **Ant** > **Runtime** > **Classpath** > **Global Entries** click **Add JARs** and select all jars in the **lib** directory



Configure the Transact Template Project

1. Unzip the **transact-project-template-x.x** file as a sub-folder of the same name in your Eclipse workspace directory (ensure you use the same workspace root directory as used above).
2. Open the **transact-project-template-x.x** project in the IDE
3. In the **Project Properties** dialog under **Java Build Path** select the **Libraries** tab and click **Add Library** to start the Add Library wizard
4. Select **User Library** then continue, then select the **transact-fluent-api-x.x** User Library before finishing the Add Library wizard
5. Click **OK** to close the **Project Properties** dialog
6. Run the **project-init** task in the Ant build file (build.xml). You will need to ensure that you have the API lib files in your Ant execution classpath.

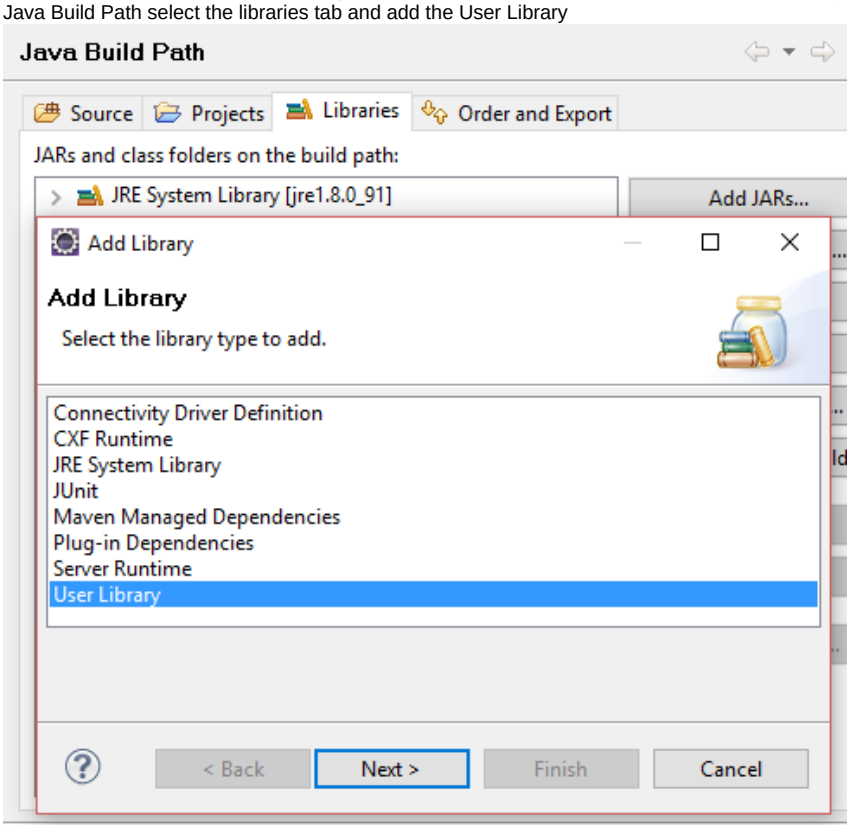
Creating new Transact Projects

You may want to try this out by following the Hello World Service Example

When starting a new development project you'll want to setup a new IntelliJ project using the steps below:

1. Copy and Paste the template project and give it the desired name (e.g. **project.myfunction**)

2. Ensure you have the **tranact-fluent-api-x.x** library in the build path. If not this can be added by opening the Project Properties dialog and under Java Build Path select the libraries tab and add the User Library



3. Open your **build.properties** file in the editor and modify the **project.name** and **project.code** properties as documented.
4. Ensure all jar files from the API project lib directory are added to the Ant execution classpath for the **build.xml** file

Generating new Groovy Services

When ever you are creating new services in the project, follow this procedure:

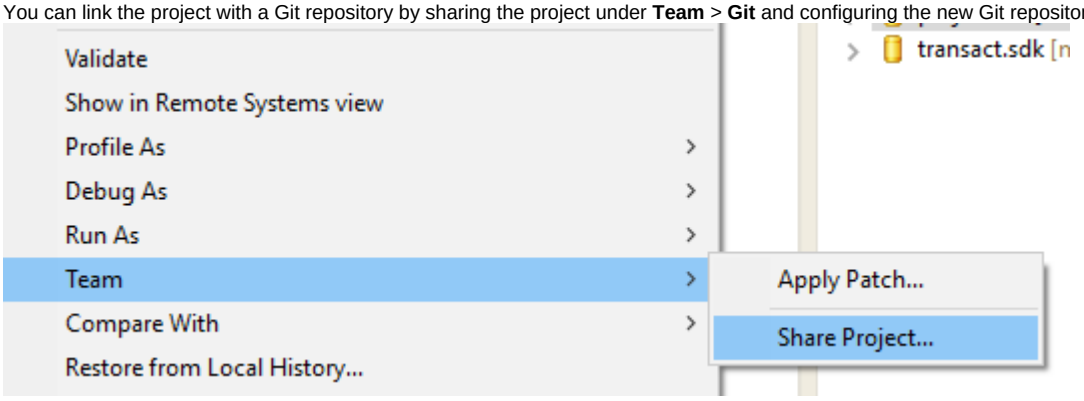
1. Open your **new-service.properties** file in the editor and modify the service generation properties as documented.
2. Execute the **generate-svc** build target.
3. Confirm that a new **build-svc-<service.code>.xml** file was created in the project root directory and that a new sub-folder was created in the **src** dire ctory containing the new service files.
4. Open your **build-svc-<service-code>.xml** file and ensure all the API jar files are in the Ant execution classpath.

You are now ready to develop and deploy your new service.

Linking your new Project to a Git Repository

Once you've created your new project you may wish to link it to a Git repository. This can be done as follows:

1. You can link the project with a Git repository by sharing the project under **Team** > **Git** and configuring the new Git repository



Cloud-Based Development in Transact Manager

Developing services in the cloud requires no setup, however the cloud development environment lacks the benefits of a modern IDE and Avoka recommends that all groovy development be done in an IDE where practical.

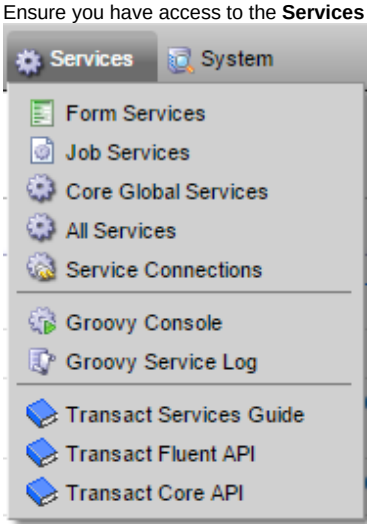
See Desktop IDE Development with The Transact Fluent API

If required, developers are able to build and maintain Groovy services in the cloud by accessing the Transact Service Container directly.

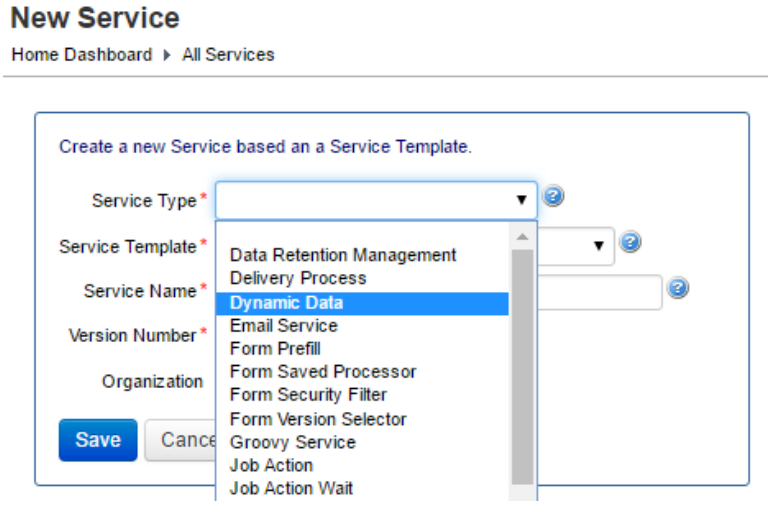
Developers are required to have a Developer account with the appropriate permissions in a non-production instance of Avoka Transact Manager. Please review the Transact Manager Administration documentation if you require information about roles and permissions.

Access to the Transact Service Container can be gained as follows:

1. Login to the non-production instance of Avoka Transact Manager
2. Ensure you have access to the **Services** menu:



3. Select the menu item that most closely represents the type of service you wish to create or **All Services** if you are not sure.
4. Click the New button to create a new service and complete the New Service wizard



5. Development of your Groovy code and unit test script can now be performed by accessing the appropriate tab of the Service Details page.

Java & Ant

If you want to quickly install Java and/or Ant without administration privileges, follow the instructions below.

Java

You can use the 'Server JRE' bundle provided in the Java download pages. This package is almost a full JDK, and is sufficient for running most Java applications, including Ant.

Download the Server JRE by following the links at http://www.oracle.com/technetwork/java/javase/downloads/index.html There are also installation instructions on that page which you should follow.

If you are using Windows, you should setup a JAVA_HOME environment variable, and adjust your PATH environment variable. A shortcut is to type Windows-R, then type "rundll32 sysdm.cpl,EditEnvironmentVariables" and enter. Edit the Path variable to add the Server JRE's 'bin' folder. Create a new variable JAVA_HOME and point it to the Server JRE location (the parent of the 'bin' folder).

Ant

Download a binary release of Ant for your platform from https://ant.apache.org/bindownload.cgi

Unzip the distribution into a folder of your choice, and edit your environment variables to adjust the PATH variable, and create a new ANT_HOME variable. Add the distribution's 'bin' folder to your PATH variable as described in the Java instructions, and create ANT_HOME to point to the unzipped Ant folder (the parent of the 'bin' folder).

Test

Test your installation by typing the commands below in a CMD or terminal prompt. You should see similar output if your setup is correct.

Commands

```
java -version
ant -version
```

Output

```
xxxxxx@XXXXXXXXXX MINGW64 /c/dev
$ java -version
java version "1.8.0_111"
Java(TM) SE Runtime Environment (build 1.8.0_111-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.111-b14, mixed mode)
xxxxxx@XXXXXXXXXX MINGW64 /c/dev
$ ant -version
Apache Ant(TM) version 1.10.1 compiled on February 2 2017
```

Hello World Service Example

In this example we will create and deploy a simple Fluent Dynamic Data service to receive a person's name as input and return a random greeting for that person.

Please ensure you have configured your development environment as described in Development Environment Setup.

> This article assumes you are using a Desktop IDE, however the process for Cloud development is very similar, albeit with fewer steps.

Watch the video

and/or follow the documented process below.

New Service Creation

To create the bones of the new Groovy service for this example please follow these steps:

1. Create your new project based on the Transact Template Project and name it **project.helloworld**. Review the relevant instructions for your IDE here - Desktop IDE Development with The Transact Fluent API.
2. Run the **scaffold** target on the **scaffold.xml** Ant build file. Ensure you are comfortable Working with the Ant Build Files first.
3. Open the **new-service.properties** file and configure the following values:

Property	Value
service.name	Welcome Greeting
service.code	welcome
service.template	Fluent Dynamic Data
service.version	1

4. Run the **create-new-service** Ant target in the **build.xml** file and verify the creation of the new **build-svc-<service.code>.xml** file and the **src/<service.code>** directory with service files.

Configure Service Parameters

This service requires a configuration parameter that contains the list of greetings which we will select from at random to generate the greeting in the response. To configure this service parameter, perform the following actions:

1. Open the service-def.json file and add a service parameter with the following attributes:

Attribute	Value
name	greetingsCsv
description	A comma separated list of greetings used to generate a greeting at random.
type	Text
required	true
readOnly	false
value	Hi,Hello,G'Day,Allo,Hola,Bonjour,Salut,Buenos dias
clearOnExport	false

2. Your JSON should look something like this:

```
{
  "name": "greetingsCsv",
  "description": "A comma separated list of greetings used to generate a greeting at random.",
  "type": "Text",
  "required": true,
  "readOnly": false,
  "value": "Hi,Hello,G'Day,Allo,Hola,Bonjour,Salut,Buenos dias",
  "clearOnExport": false
}
```

3. Ensure you've added a comma between this and other service parameters in the list, then save the file. For more information on these configuration options see Configuring Service Definitions with service-def.json.

Groovy Scripting

The generated Groovy script for the newly created service should be found in the **src/welcome** directory in a file called **WelcomeGreeting.groovy**. You will open this file in your IDE to edit the script.

In the Groovy script for this service we want to perform the following functions:

1. Declare and read an input parameter called **name**. If no input was provided then default to 'friend'.
2. Select a greeting at random from the **greetingsCsv** service parameter
3. Create a result **Map** containing the greeting and convert it to JSON data before returning it

Feel free to try this yourself, or paste the following code into your Groovy script:

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.vo.*
import javax.servlet.http.*
import groovy.json.JsonBuilder

class WelcomeGreeting {

    String invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {

        // Read the persons name from the request
        String personsName = request.getParameter('name')?:'friend'

        // Select a greeting from the list at random
        List<String> greetings = svcDef.paramsMap.greetingsCsv.tokenize(',')
        String randomGreeting = greetings.get(Math.floor(Math.random() * greetings.size()).intValue())

        // Create the result Map containing the greeting
        Map result = [:]
        result.greeting = (randomGreeting + ' ' + personsName)

        // Convert to JSON and return
        return new JsonBuilder(result).toString()
    }
}
```

Take care not to modify the name or attribute list of the *invoke* method. This method signature must remain unchanged for it to be located and executed by the service container.

Unit Testing

In order to ensure that our Groovy script is working correctly we can write a simple unit test to check its function under a range of scenarios. Unit test scripts are also Groovy scripts themselves. The generated Unit Test Script for the newly created service will be found in the same directory in a file called **Welcome GreetingTest.groovy**.

In this Unit Test Script we want to test our service under 2 scenarios:

1. Where no input name is provided we should receive a greeting containing the word 'friend'.
2. Where a name is provided then we should receive a greeting containing that name.

You can attempt this yourself, or paste the following Groovy code into your Unit Test Script:

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

class WelcomeGreetingTest {

    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        MockRequest request = new MockRequest()
        Map params = [
            "svcDef": svcDef,
            "txn": new MockVoBuilder().createTxnOpened(),
            "request": request,
            "user": null
        ]

        // Scenario 1: No name parameter provided
        String result = (String) new ServiceInvoker(svcDef).invoke(params)
```

```
        logger.info result
        String greeting = (String)(new Path(result).val("greeting"))
        assert greeting.endsWith("friend")

        // Scenario 2: Name provided in request
        request.setParameter("name", "Johny")
        result = (String) new ServiceInvoker(svcDef).invoke(params)
        logger.info result
        greeting = (String)(new Path(result).val("greeting"))
        assert greeting.endsWith("Johny")
    }
}
```

Deploy and Test the Service

Execution of Groovy services must occur in the Transact Manager Groovy Service Container. Deployment and testing of Groovy services in the Transact Manager environment is facilitated by the service build file and each service will have it's own build file that contains a range of targets related to these activities.

For more details on the Ant targets contained in the build files refer to Working with the Ant Build Files.

In order to deploy the service to Transact Manager you need to identify which environment you want to deploy to and the credentials that you want to use. To do this you need to edit the **transact-auth.properties** file and complete the required information.

Do not use quotes to delimit property values - use naked strings only:

- **manager.username=johny** - Correct
- **manager.username="johny"** - Incorrect

Property	Description
manager.url	The URL to the Transact Manager console e.g. https://dev.transact.yourdomain.com/manager.
manager.username	The authorized user account on Transact Manager to use for deploying the service.
manager.password	The password for the authorized user account on Transact Manager
manager.clientCode	The client code can be via on the **Organizations** menu item under the **Forms** menu

For the purposes of this exercise you will just need to run the default **build** target of the **build-svc-welcome.xml** file which will perform all deploy and test actions in sequence.

To execute this build target you must ensure that all jars in the lib folder of the **transact-fluent-api-x.x** project is in the Ant execution classpath. Check the Desktop IDE Configuration Instructions if you are unsure how to do this.

If your service is written correctly and the unit tests pass you should see output similar to the following as a result of your Ant build execution:

```
Request: POST https://mydomain.com/manager/secure/rest/service-definitions/v1/run-unit-test/ HTTP/1.1
JUnit XML Report: C:\Workspaces\Intellij\project.helloworld\out\test-welcome.xml
Success: {
  "serviceName": "Welcome Greeting",
  "versionNumber": 1,
  "clientCode": "maguire",
  "testStatus": "Success",
  "message": "Test succeeded in 266 ms",
  "logger": "22:26:56,892 INFO  {\"greeting\":\"Bonjour friend\"}\n22:26:56,899 INFO  {\"greeting\":\"Hello
Johny\"}"
}
GroovyLogger:
-------------
22:26:56,892 INFO  {"greeting":"Bonjour friend"}
22:26:56,899 INFO  {"greeting":"Hello Johny"}
build
Ant build completed successfully in 5s at 31/10/2016 10:26 PM
```

Note the line advising the creation of a JUnit XML Report which may be opened to review the test results.

Groovy Script Annotations

The Transact Fluent SDK includes a document generation facility that interprets annotations in the Groovy Script to automatically produce template based help documentation to describe the service. For details of these annotations please review the ServiceDoc Annotations page.

The code below demonstrates the use of annotations to document the service used in this example:

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.vo.*
import javax.servlet.http.*
import groovy.json.JsonBuilder

/**
 This service receives a persons name and generates a random greeting for that person in either French, English
or Spanish.

 @since Manager 5.0

 @sample Request
 <pre> {<br>
  "name": "Johny"<br>
 }</pre>

 @sample Response
 <pre> {<br>
  "greeting": "Hola Johny"<br>
 }</pre>

 */
class WelcomeGreeting {

    String invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {

        // Read the persons name from the request
        /* @in name The name of the person receiving the greeting; defaults to 'friend' */
        String personsName = request.getParameter('name')?:'friend'

        // Select a greeting from the list at random
        List<String> greetings = svcDef.paramsMap.greetingsCsv.tokenize(',')
        String randomGreeting = greetings.get(Math.floor(Math.random() * greetings.size()).intValue())

        // Create the result Map containing the greeting
        Map result = [:]
        /* @out greeting The generated greeting string E.g. Bonjour Johny */
        result.greeting = (randomGreeting + ' ' + personsName)

        // Convert to JSON and return
        return new JsonBuilder(result).toString()
    }
}
```

After redeploying the annotated Groovy service in Transact Manager you should now see some useful content on the Help Doc tab of the Service record:

Welcome Greeting - v1 (maguire)

Home Dashboard ▸ Form Services ▸ Service Definition

Service Definition | Groovy Script | Parameters Edit | Parameters | Unit Test | Test Parameters | Groovy Service Log

Welcome Greeting

This service receives a persons name and generates a random greeting for that person in either French, English or Spanis

Compatibility Notes

Module	Compatibility
Manager	5.0

Service Parameters

The following configuration parameters are available in the Parameters tab of the Service Definition:

Parameter Name	Description
greetingsCsv	Required: A comma separated list of greetings used to generate a greeting at random.

Input Variables

The following input variables are available to the consumer of this service when making calls

Variable Name	Description
name	The name of the person receiving the greeting; defaults to 'friend'

Response Data

The following table details the response data that is available to the consumer of this service when making calls:

Data Element	Description
greeting	The generated greeting string E.g. Bonjour Johny

Examples

The following table provides some examples to demonstrate usage scenarios:

Example	Description

Request	<pre>{
	"name": "Johny"
	}</pre>

Response	<pre>{
	"greeting": "Hola Johny"
	}</pre>

Groovy Language

Why Groovy?

Groovy is a powerful scripting language which runs on the Java Virtual Machine. This gives you access to all the facilities provided by Transact Manager and Java, while providing a scripting language without the steep learning curve of Java.

You can think of Groovy as JavaScript for the server. It is a dynamic language, there are no compile and deploy steps. Just write it and run it.

Avoka Transact provides customized Groovy runtime with security constraints for running in an application server context. For Groovy version information see 'Groovy Lang' under 3rd Party Java Libraries.

Groovy also makes it easy to produce and consume XML and Json format data, which are very common data exchange formats, particularly for exchanging data with web services.

Simple Example

The simple hello world example below defines a variable called name, and assigns it the value 'World'. Then on the next line this variable $name is included in the Hello text and this line is printed using the println function.

```
def name = 'World'
println "Hello $name!"
```

And as expected the result is:

```
Hello World!
```

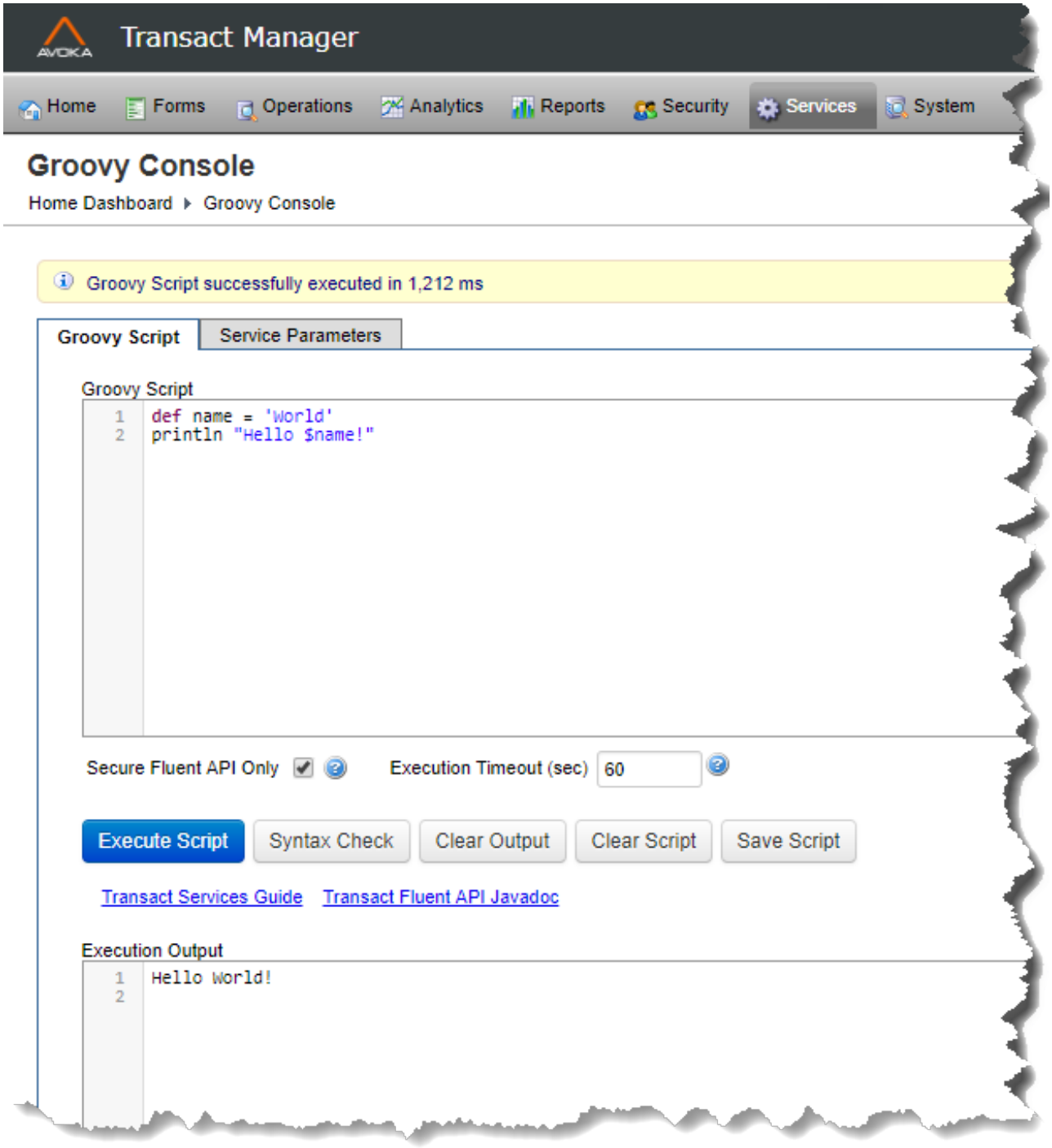
Learning Groovy

If you are familiar with programming in a modern language then you will be able to quickly learn Groovy. It has a C like syntax in common with C#, JavaScript and Java, but has scripting features so you don't have to write lots of code.

This Groovy Guide will provide you with a quick and gental introduction to the language, but will give you the skills you need to develop Groovy Service scripts.

As you are working through the Groovy Guide examples, please use the Groovy Script Console under the Services menu. This provides you a play ground to try out examples and prototype Groovy scripts.

Here you can see a simple script and its output that was written and executed in the Groovy Console:



Script Type Checking

Scripts you develop in Transact Manager are static type checked for security and performance reasons. This means you cannot use the dynamic type features of Groovy in Transact Manager. Static type checking enforces safer coding practices. You can discover more about this topic at Static Compilation & Type Checking

Next Steps

The next topic discusses Groovy Declarations

Once you have completed this Groovy Guide and you want to learn more please see the Groovy Language Documentation.

Groovy Declarations

This section discusses some of the key Groovy script declarations.

Please review our best practice recommendation regarding Using Defined Type Declarations in Favor of Dynamic Types

Comments

Groovy uses C style code comments, for specifying content which should not be executed.

```
// This is a one line comment
def a = 1 + 1

/*
 This is a multi line
 comment block.
*/
def b = 2 + 2
```

Variables

To declare a variable in Groovy simply prefix it with the def statement.

```
// Defining a string variable
def a = "Hello world"
```

The Groovy runtime will attempt to figure out what type to use when you define a variable. For example:

```
def var1 = 'World'
println var1.class.simpleName

def var2 = true
println var2.class.simpleName

def var3 = 1
println var3.class.simpleName

def var4 = 3.1425
println var4.class.simpleName
```

This will print out:

```
String
Boolean
Integer
BigDecimal
```

Strings

Text values are provided using the String class. String values can be created using single or double quotes and contain C style literals or control characters.

```
def msg1 = "Hello World"

def msg2 = 'Hello World'
```

Multi line text values can be created using triple quotes, for example:

```
def longMsg = """
Line 1
Line 2
Line 3
"""

println longMsg
```

WARNING: When splitting up multi-line strings you can't simply have a line return and then continue on the next line because Groovy interprets each line separately. For example:

```
// This will NOT work
def longMsg = "abcefchijklmn"
      + "opqrstuvwzyz"

// Groovy actually interprets this as two expressions and adds a semi-colon at the end of each line,
// which leave the second line hanging on its own
def longMsg = "abcefchijklmn";
      + "opqrstuvwzyz"

// You can use parentheses to prevent this as below
def longMsg = ("abcefchijklmn"
      + "opqrstuvwzyz")
```

String Templating

Groovy also provides a GString class for doing string templating or interpolation. GStrings can be very useful for creating JSON or XML message bodies.

```
def accountName = ...
def ruleset = ...
def transID = ...

def message = '''
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body></SOAP-ENV:Body>
        <m:transaction-continue xmlns:m="java:com.verid.carbon.integration.datatypes">
            <settings>
                <account-name>$accountName</account-name>
                <mode>$mode</mode>
                <ruleset>$ruleset</ruleset>
                <transaction-id>$transID</transaction-id>
            </settings>
        </m:transaction-continue>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
'''
```

For larger blocks of formatted text such as HTML Email message bodies we often use Apache Velocity for templating operations. Velocity provides a powerful templating capabilities and is useful for keeping large amounts of formatted text out of your code.

In the example below the message Velocity template is provides as an external service parameter. This value is used to create a Velocity VelTemplate which is then merged with the model map to create the SOAP message.

```
import com.avoka.tm.util.VelTemplate

def model = [:]
model.accountName = ...
model.ruleset = ...
model.transID = ...

def message = new VelTemplate().setTemplate(serviceParameters.messageTemplate).setModel(model).merge()
```

Dates

Date and time values are provided using the Date class. This class provides convenient methods for formatting date strings and parsing string values into date objects. These formatting and parsing patterns use the SimpleDateFormat

```
// Create a new date and print it out
def now = new Date()
println now

// Display value using ISO date format
println now.format("yyyy-MM-dd")

// Display value using ISO8601 date time format
println now.format("yyyy-MM-dd'T'HH:mm:ss")
```

This script will display something like:

```
Mon Jul 22 21:26:51 EST 2013
2013-07-22
2013-07-22T21:26:51
```

Please note when calling some TM Java APIs you may need to explicity define java.util.Date types if the Groovy Runtime does not resolve the API method correctly.

Collections

When working with Groovy scripts we are often dealing with collections objects. Groovy provides convenient methods working with collections. Please also see the Groovy Collections topic.

Maps / Dictionaries

Maps or dictionaries can be created and accessed using the following syntax.

```
// Define a new map
def car = [:]

// Set values using dictionary style access
car["seats"] = 6
car["color"] = "blue"
car["year"] = 2013

// Read values using dictionarly style access
println car["seats"]
println car["color"]
println car["year"]

// Set value using property style access
car.seats = 6
car.color = "blue"
car.year = 2013

// Read value using property style access
println car.seats
println car.color
println car.year

// Test whether map has an entry accessed using the key "year"
println car.containsKey("year")

// Test whether map has an entry value of 2015
println car.containsValue(2015)
```

You can loop through the entries of a map using the following syntax.

```
// Define map and assign values
def car = [ "seats":6, "color":"blue", "year":2013 ]

for (entry in car) {
   println entry.key + " : " + entry.value
}
```

Which will print out:

```
seats : 6
color : blue
year : 2013
```

Under the covers Groovy uses Java LinkedHashMap for maps. This class support key based access and insertion order iteration.

Lists / Arrays

Lists or arrays are collections with a series of elements which can be iterated through. The syntax for creating lists is very straightforward.

```
// Create a list of fruit
def fruitSalad = [ "banana", "apple", "orange", "mango", "grapes", "kiwi fruit" ]

println "ingredients: " + fruitSalad
```

```
println "number of items: " + fruitSalad.size
for (item in fruitSalad) {
        println "\t" + item
}

// Modify list
fruitSalad[0] = "passion fruit"
fruitSalad.remove("apple")
fruitSalad.add("stawberry")
println "ingredients: " + fruitSalad

// Test whether the list contains a "apple" value
println fruitSalad.contains("apple")
```

This script will print out the results:

```
ingredients: [banana, apple, orange, mango, grapes, kiwi fruit]
number of items: 6
        banana
        apple
        orange
        mango
        grapes
        kiwi fruit
ingredients: [passion fruit, orange, mango, grapes, kiwi fruit, stawberry]
false
```

Under the covers Groovy uses Java ArrayList for lists. This class support very fast access and array additions.

The next topic deals with Groovy Control Statements.

Control Statements

Groovy control statements are modelled on C style statements. You can do a lot with very little.

Branching Logic

Groovy support if/else statements, switch statements and the ternary operator. Generally you should be able to get by with using if/else statements.

If Else Statements

The if else statements are pretty straightforward.

```
def seats = 2

if (seats == 0) {
  println 'There are no seats'

} else if (seats == 1) {
  println 'There is one seat'

} else if (seats == 2) {
  println 'There are two seats'

} else {
  println 'There are more than two seats, if you think positively'

}
```

which will print out the message

```
There are two seats
```

Compact If Else plus Assignment

There is also the ternary operator, which provides an if else statement combined with a variable assignment.

```
def seats = 1
def msg = (seats == 2) ? "There are two seats left" : "Sorry we don't have 2 seats left"
println msg
```

This script will print out the message

```
Sorry we don't have 2 seats left
```

The Elvis Operator ?:

The Elvis operator is a shortening of the ternary operator and is very convenient for assigning to a variable a sensible default value where it would otherwise be set to null. In the code examples below the ternary operator requires you to repeat the value you want to assign if the value tested is not null or false, where the elvis operator will use the value tested if it is not null or false.

```
// Ternary operator
firstName = request.getParameter('title') ? request.getParameter('title') : ''
// Elvis operator
firstName = request.getParameter('title') ?: ''
```

Looping

For Statement

Using the for statement you can loop over a range of numbers, lists and maps.

Number Iteration

In a for statement you can specify the starting index, the iteration test and increment operation.

```
def items = [ "banana", "apple", "orange", "mango", "grapes", "kiwi fruit" ]

// Start at index 0, only loop while the index is less than the length of the list,
// and increment the index by one for each iteration
for (int i = 0; i < items.size(); i++) {
   println i + ". " + items[i]
}
```

This script will print out the message

```
0. banana
1. apple
2. orange
3. mango
4. grapes
5. kiwi fruit
```

List Iteration

When looping through a list you can use the for each item in items syntax

```
def items = [ "banana", "apple", "orange", "mango", "grapes", "kiwi fruit" ]

// For each item in the list of items
for (item in items) {
   println item
}
```

This script will print out the message

```
banana
apple
orange
mango
grapes
kiwi fruit
```

Map Iteration

Maps are special in that you can iterate through them a number of ways.

```
def car = [ "seats" : 4, doors: 2, "color" : "red", "year" : 2011 ]

// For each map entry
for (entry in car) {
   println entry.key + ": " + entry.value
}
```

```
def car = [ "seats" : 4, doors: 2, "color" : "red", "year" : 2011 ]

// For each map entry key (name)
for (key in car.keySet()) {
   println key + ": " + car[key]
}
```

```
def car = [ "seats" : 4, doors: 2, "color" : "red", "year" : 2011 ]

// For each value in the map
for (value in car.values()) {
   println value
}
```

Please note to find something in a map you can generally lookup the value using the key.

Returning Values

In Groovy script you can return an object to the calling code using the return statement.

```
def car = [ "seats" : 4, doors: 2, "color" : "red", "year" : 2011 ]

if (car["doors"] == 2) {
   return "We don't insure sports cars"
}

println 'Continue processing'

// Continue processing
..
```

Catching Errors

If you need to handle errors and continue processing you can use the try / catch statement.

```
def textValue = "z123"

def intValue
try {
   intValue = Integer.parseInt(textValue)

} catch (Exception e) {
   println e
   intValue = -1
}

println "Integer value: " + intValue
```

Configuring Service Definitions with service-def.json

The key files that define a Fluent Groovy Service are:

- **MyService.groovy**: The Groovy code that executes when the service is called.
- **MyServiceTest.groovy**: A test script that is used to verify the function of the Groovy service.
- **service-help.html**: An html help document describing the usage and configuration options of the service. Note: with the Transact Fluent SDK, this document is auto-generated based on annotations in the Groovy service itself.
- **service-def.json**: A JSON file that defines the attributes and configuration parameters for the Groovy service

Note: these 4 files are created for you when you run the **create-new-service** target in your project build file so there is no need to create them from scratch. See Working with the Ant Build Files.

This article looks specifically at the **service-def.json** file and how it is used.

You can find additional information in the **readme.html** file within the **transact-fluent-api-x.x** project (see Development Environment Setup).

Sample Service Definition

When you generate a service using the SDK build files, a **service-def.json** file will be created for you that looks something like this:

```
{
  "name": "Welcome Greeting",
  "description": "Generates a random using multiple languages",
  "type": "Dynamic Data",
  "version": 1,
  "tmMinVersion": "5.0.0",
  "serviceConnection": "Language Translation",
  "parameters": [
    {
      "name": "groovyScript",
      "filePath": "WelcomeGreeting.groovy",
      "fileIncludes": [],
      "bind": true,
      "required": false,
      "clearOnExport": false,
      "readOnly": false
    },
    {
      "name": "Unit Test Script",
      "filePath": "WelcomeGreetingTest.groovy",
      "fileIncludes": [],
      "bind": false,
      "required": false,
      "clearOnExport": false,
      "readOnly": false,
      "unitTest": true
    },
    {
      "name": "Help Doc",
      "type": "HTML",
      "filePath": "service-help.html",
      "bind": false,
      "required": false,
      "clearOnExport": false,
      "readOnly": false
    }
  ]
}
```

From this sample you can see that the JSON structure consists of some service level attributes as well as an array of parameters with their own configurations.

The 3 default parameter definitions identify the other 3 files that define the service, being the Groovy script, the test script and the help document.

Note: at the time of import Transact Manager identifies these 3 key parameters by name and treats them differently to other parameters so it is important that you do not modify the **name** attribute from their initialized values.

Service Level Configuration

The following configuration attributes are available for use at the top level and will have effect at the service level:

Attribute	Description

name	The service name is a short succinct label for the service and is initialized with the **service.name** property defined in the **new-service.properties** file when you generate your service. This name must be unique in the Transact Manager organization that it is deployed to and is used as the identifier when calling a service.
description	A one line description of the service to provide a brief overview of the service function and purpose.
type	See Transact Service Types. This value is initialized based on the **service.type** property defined in the **new-service.properties** file when you generate your service. For a full list of supported service.type options see
version	The service version is an integer value starting at 1 and can be incremented to deploy a new version of your service while keeping the old version on the server.
tmMinVersion	Defines the minimum Transaction Manager (TM) version required to use this service. This is a 3 level version in the format **5.x.x** and should match the TM version shown in the **System** >> **About** page in the TM Administration Console. During deployment of this service, the server version will be checked for compatibility.
serviceConnection	Optionally you may specify a Service Connection that is used by this service. This value should contain the service connection **name**. During deployment of this service, if the referenced service connection does not exist on the server it will be created with blank configuration. If you are connecting to an external API during the execution of your service it is important to externalize your connection endpoint and credentials in a Service Connection, rather than making these service parameters. See also: Managing multiple service endpoints and credentials for external service calls

Parameter Configuration

Each service will have some configuration attributes associated with it. These should be externalized from the Groovy script into Service Parameters to enable users of the service to configure these attributes without having to modify groovy script. Give consideration to which attributes should be made Service Parameters for the services you produce.

When configuring service parameters on your service there are a number of attributes that you can use to effect different behaviors. The attributes specified below can be used in the service-def.json file used in the Groovy Ant Tasks.

Attribute	Description
name	See naming conventions - use camel case
description	The description appears in the parameters edit page as popup help
type	Supported types include *Boolean, Date, Email, Groovy Script, HTML, JSON, List, Number, Password, String*
required	*true|false* - If required is set to true then the execution framework will ensure that this parameter has a value specified against it before allowing execution of the script. If no value is set then an exception is thrown and logged.
value	The default value of the parameter. • For Date types the value should be specified in *yyyy-MM-dd* format (see Date Formats above) • List values should be specified in the format *value1:label1|value2:label2* • Boolean values provided as *true|false*
clearOnExport	*true|false* - If clearOnExport is set to true then any value configured against this parameter will be excluded when you export the service from Transact Manager. This is particularly handy when you have parameters that are not expected to be the same in different environments. This is important if you have environment specific details such as account IDs, GUIDs etc.. Note that if this attribute also effects the import behavior and could be considered a 'preserve on import' configuration as well. If set to true then any values specified with the value attribute (above) will also be ignored when importing or deploying your service to Transact Manager - this is handy because you can set the value of a parameter in Transact Manager and know that it will not be overwritten when you next deploy your service.
filePath	Used to specify the location of the relevant external file for the following parameter types: • The Groovy Script parameter • The Groovy Test parameter • Parameters of type HTML or JSON

fileIncludes	Used for to include additional external Groovy files in the Groovy Script and Unit Test Script parameters.
	
	ⓘ Note the path should be relative to the service-def.json file location so you should use the double dot notation to traverse up directory structures if required.
	
	For example, to include 2 Groovy class files from the sources root folder in my Groovy script I would use the following configuration:
	
	`{`
	`  "name": "groovyScript",`
	`  "filePath": "service-rule.groovy",`

	<pre>"fileIncludes": [
	"../Constants.groovy",
	"../Util.groovy"
	]
	}</pre>

unitTest	*true*|*false* - Controls if this parameter is a standard service parameter or a test parameter. If unitTest is set to true then this parameter will be passed to the Groovy unit test script and will not be available in the Groovy service implementation along with other standard service parameters.
readOnly	*true*|*false* - If readOnly is set to **true** then the service parameter will not be editable via Transact Manager; otherwise it will be editable. Default: **false**.

Groovy Execution Environment

Fluent SDK Security Configuration

Transact Fluent Groovy services are compiled and executed using a security configuration to protect the integrity and data security of the system.

The main components of the security system include:

- Secure Java Package Whitelist
- Groovy Static Compilation
- Illegal Token Blocking
- Client Data Access Security Context

Secure Java Package Whitelist

Fluent Groovy services and unit tests can only access Java package contained in the approved white list. Access to classes outside of the approved package whitelist is prohibited.

Individual classes in the package whitelist may still be prohibited if they are included in the blacklist.

Java Package Import Whitelist

```
com.amazonaws
com.amazonaws.auth
com.amazonaws.auth.policy
com.amazonaws.auth.presign
com.amazonaws.auth.profile
com.amazonaws.client
com.amazonaws.client.builder
com.amazonaws.services.cloudtrail
com.amazonaws.services.cloudtrail.model
com.amazonaws.services.dynamodbv2
com.amazonaws.services.dynamodbv2.datamodeling
com.amazonaws.services.dynamodbv2.datamodeling.marshallers
com.amazonaws.services.dynamodbv2.datamodeling.unmarshallers
com.amazonaws.services.dynamodbv2.document
com.amazonaws.services.dynamodbv2.document.api
com.amazonaws.services.dynamodbv2.document.spec
com.amazonaws.services.dynamodbv2.document.utils
com.amazonaws.services.s3
com.amazonaws.services.s3.event
com.amazonaws.services.s3.iterable
com.amazonaws.services.s3.model
com.amazonaws.services.s3.transfer
com.amazonaws.services.s3.transfer.exception
com.amazonaws.services.s3.transfer.model
com.amazonaws.services.securitytoken
com.amazonaws.services.securitytoken.model
com.amazonaws.services.securitytoken.model.transform
com.amazonaws.services.sns
com.amazonaws.services.sns.model
com.amazonaws.services.sns.util
com.amazonaws.services.sqs
com.amazonaws.services.sqs.buffered
com.amazonaws.services.sqs.model
com.amazonaws.util
com.amazonaws.util.json
com.avoka.component.docusign
com.avoka.component.sharepoint
com.avoka.component.sharepoint.service
com.avoka.component.sharepoint.type
com.avoka.core.groovy
com.avoka.tm.http
com.avoka.tm.job
com.avoka.tm.query
com.avoka.tm.svc
com.avoka.tm.test
com.avoka.tm.util
com.avoka.tm.vo
com.auth0.jwt
com.auth0.jwt.pem
com.fasterxml.jackson.dataformat.xml.annotation
com.google.gson
com.google.gson.annotations
com.google.gson.reflect
com.google.gson.stream
com.itextpdf.text.pdf
com.jcraft.jsch
eu.bitwalker.useragentutils
```

```
java.io
java.lang
java.math
java.net
java.nio
java.nio.charset
java.security
java.security.acl
java.security.cert
java.security.interfaces
java.security.spec
java.sql
java.text
java.time
java.time.chrono
java.time.format
java.time.temporal
java.time.zone
java.util
java.util.jar
java.util.logging
java.util.prefs
java.util.regex
java.util.zip
javax.crypto
javax.crypto.interfaces
javax.crypto.spec
javax.mail
javax.mail.internet
javax.mail.util
javax.net.ssl
javax.security.auth
javax.security.auth.callback
javax.security.auth.kerberos
javax.security.auth.login
javax.security.auth.spi
javax.security.auth.x500
javax.security.cert
javax.security.sasl
javax.sql.rowset
javax.sql.rowset.serial
javax.sql.rowset.spi
javax.servlet.http
javax.xml
javax.xml.bind
javax.xml.crypto
javax.xml.parsers
javax.xml.soap
javax.xml.stream
javax.xml.transform
javax.xml.validation
javax.xml.ws
javax.xml.xpath
org.apache.xerces.dom
groovy.json
groovy.sql
groovy.text
groovy.time
groovy.util
groovy.xml
org.apache.commons.codec
org.apache.commons.codec.binary
org.apache.commons.codec.digest
org.apache.commons.codec.language
org.apache.commons.codec.language.bm
org.apache.commons.codec.net
org.apache.commons.fileupload
org.apache.commons.io
org.apache.commons.io.comparator
org.apache.commons.io.filefilter
org.apache.commons.io.input
org.apache.commons.io.monitor
org.apache.commons.io.output
org.apache.commons.lang3
org.apache.commons.lang3.builder
org.apache.commons.lang3.concurrent
org.apache.commons.lang3.exception
org.apache.commons.lang3.math
```

```
org.apache.commons.lang3.mutable
org.apache.commons.lang3.text
org.apache.commons.lang3.text.translate
org.apache.commons.lang3.time
org.apache.commons.lang3.tuple
org.apache.http
org.apache.http.auth
org.apache.http.client
org.apache.http.client.config
org.apache.http.client.entity
org.apache.http.client.methods
org.apache.http.client.protocol
org.apache.http.client.utils
org.apache.http.conn
org.apache.http.conn.routing
org.apache.http.conn.socket
org.apache.http.conn.ssl
org.apache.http.conn.util
org.apache.http.cookie
org.apache.http.entity
org.apache.http.impl.auth
org.apache.http.impl.client
org.apache.http.impl.conn
org.apache.http.impl.cookie
org.apache.http.impl.execchain
org.apache.http.io
org.apache.http.message
org.apache.http.params
org.apache.http.pool
org.apache.http.protocol
org.apache.http.ssl
org.apache.http.util
org.joda.time
org.joda.time.base
org.joda.time.chrono
org.joda.time.convert
org.joda.time.field
org.joda.time.format
org.joda.time.tz
org.w3c.dom
org.w3c.dom.bootstrap
org.w3c.dom.events
org.w3c.dom.ls
org.xml.sax
org.xml.sax.ext
org.xml.sax.helpers
```

The following individual classes are also allowed to be referenced.

Additional Java Class Whitelist

```
com.avoka.component.xml.DropDownListUtils
com.avoka.core.groovy.GroovyLogger
com.avoka.core.groovy.runtime.GroovyScriptContext
groovy.lang.GString
groovy.ui.SystemOutputInterceptor
org.apache.click.servlet.MockRequest
org.apache.xerces.dom.DeferredDocumentImpl
```

In addition, the following packages are in the whitelist for unit test code only:

Unit Test Package Whitelist

```
org.junit
org.junit.rules
org.junit.runner
org.junit.runners
org.mockito
org.mockito.configuration
org.mockito.exceptions.base
org.mockito.exceptions.misusing
org.mockito.exceptions.stacktrace
org.mockito.exceptions.verification
org.mockito.exceptions.verification.junit
```

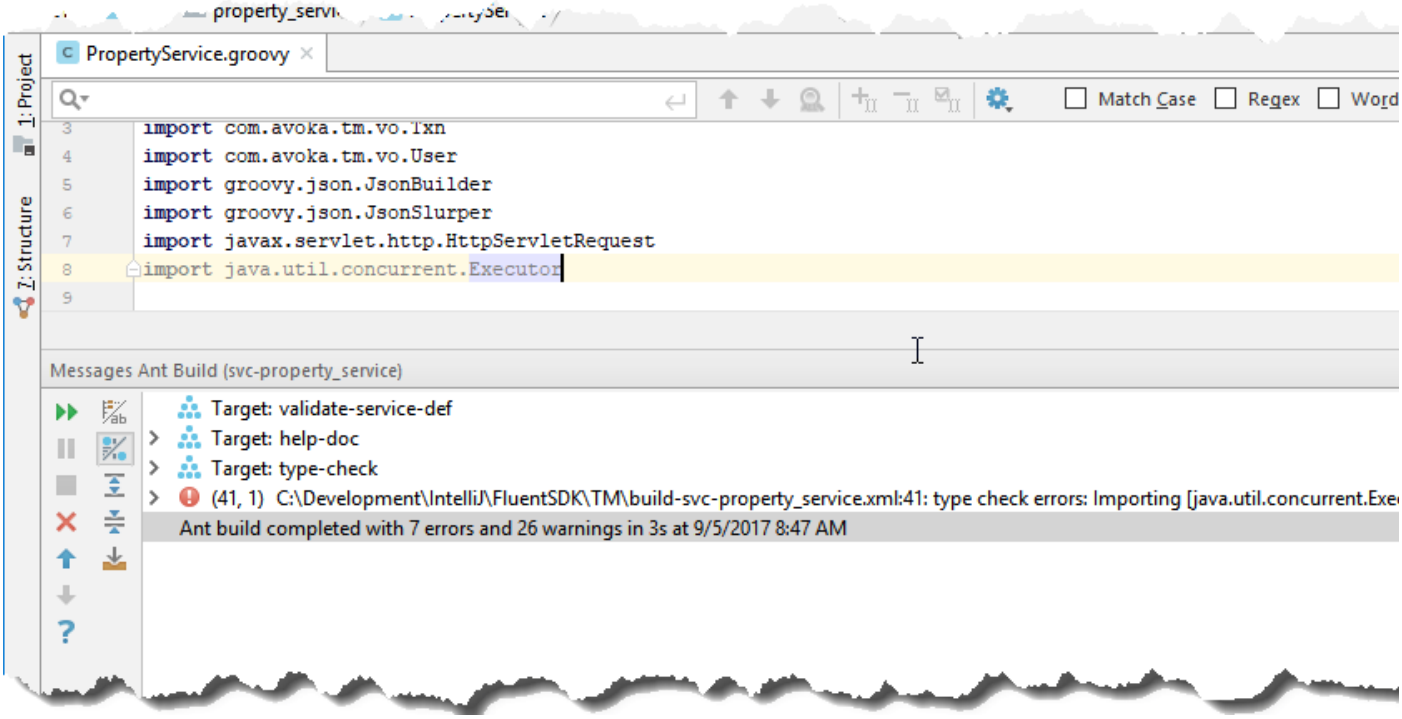
```
org.mockito.hamcrest
org.mockito.invocation
org.mockito.junit
org.mockito.listeners
org.mockito.mock
org.mockito.plugins
org.mockito.quality
org.mockito.runners
org.mockito.session
org.mockito.stubbing
org.mockito.verification
```

If you need changes to the package whitelists please contact Avoka support for assistance.

Security Error Examples

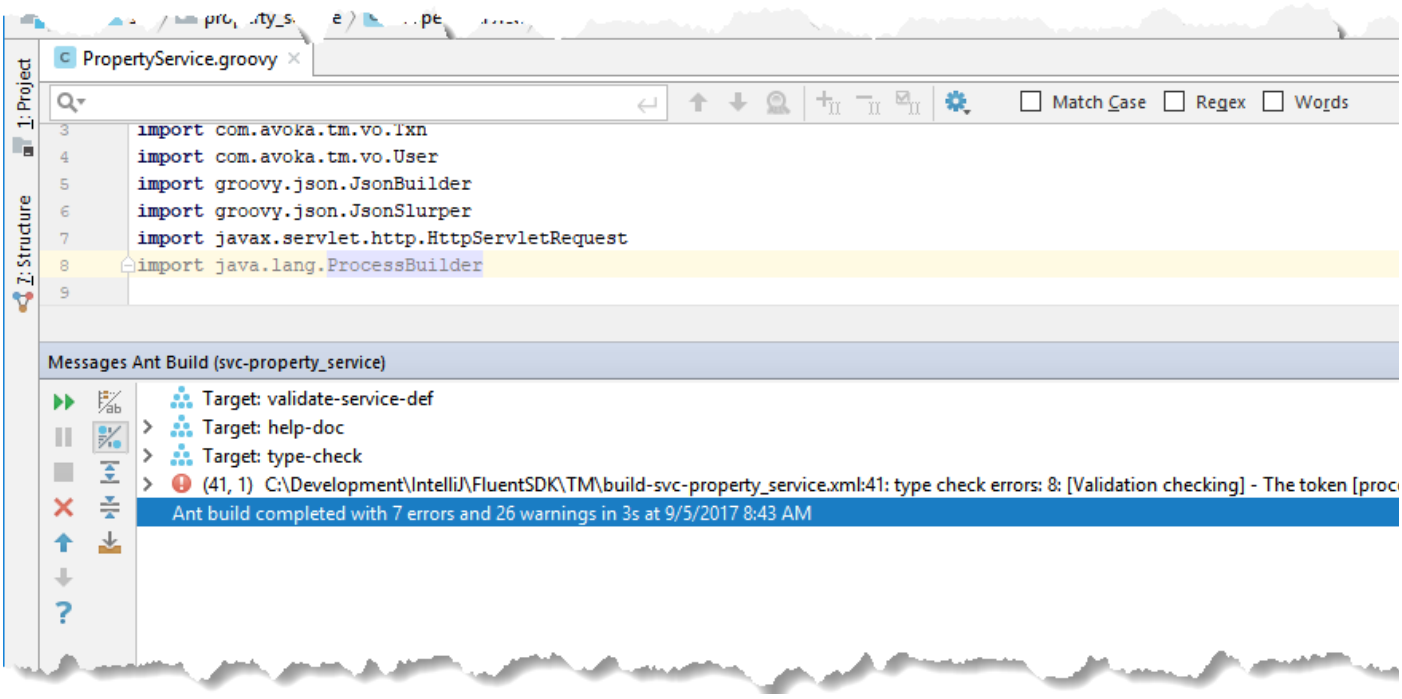
Import Not Allowed Example

This examples shows attempted import of a class in the IntelliJ IDE, which is not in the whitelist.



Illegal Token Example

This examples shows usage of blacklisted token in the IntelliJ IDE which cannot be used in a service.



Runtime Security

The Groovy Script runtime environment has some restrictions to prevent the execution of code which could cause system integrity issues or result in access to unauthorized data. If the runtime determines the script is performing an illegal operation the script will not be executed an error message:

```
Groovy Script contains an illegal reference
```

If the script attempts to access data which it is not authorized to then a null value will be returned.

Most Groovy Services are executed in a client Organization context which restricted access to Transaction data. Only system administrators with explicitly configured global access policies can access multiple organizations transaction data.

Any Groovy Script which is executed in the Groovy Console will be logged in the Security Audit Log along with details of the user and when this was performed.

Please also note the execution of Groovy Services is logged in the database Groovy Service Log, and changes to Service Paramters are logged to the Security Audit Log.

Static Compilation & Type Checking

In order to enforce a Groovy execution environment with high performance and strong security controls, Fluent Groovy services deployed into Transact are compiled statically. In addition to providing a much stronger security model, statically compiled Groovy services also execute at near native Java speed, much faster than dynamically compiled Groovy. One difference with statically compiled Groovy services is that you cannot use some of Groovy's dynamic language features such as GPath expressions or Groovy meta programming features.

If you are used to writing Java code, this will be a familiar coding environment for you. If you have been developing Groovy scripts to run in a dynamic compilation environment then this article may help you make the adjustments required to transition to static Groovy compilation. Documented below are some common issues found when you move from dynamic Groovy to static Groovy:

GPath Expressions

GPath is a path expression language integrated into Groovy which allows parts of nested structured data to be identified. This feature is often used in the context of processing XML but also for standard object graphs or nested Map structures.

As GPath expressions are not supported in static Groovy, the Transact Fluent SDK includes a very handy utility class (com.avoka.tm.util.Path) that can be used in place of GPath expressions. See also:

- Interrogating Structured Data Paths

Multiple Assignments

Groovy allows you to perform multiple assignments in a single statement. E.g.

```
// Break out the elements of the date of birth
def (dobDay, dobMonth, dobYear) = dobStr?.tokenize('/')
```

Static type checking will not allow this and will give the following error message:

- [Static type checking] - Multiple assignments without list expressions on the right hand side are unsupported in static type checking mode

To overcome this issue you must resort to more traditional assignment methods:

```
// Break out the elements of the date of birth
String dobDay, dobMonth, dobYear
String[] dobItems = dobStr?.tokenize('/')
if(dobItems.length == 2){
    dobDay = dobItems[0]
    dobMonth = dobItems[1]
    dobYear = dobItems[2]
}
```

See also:

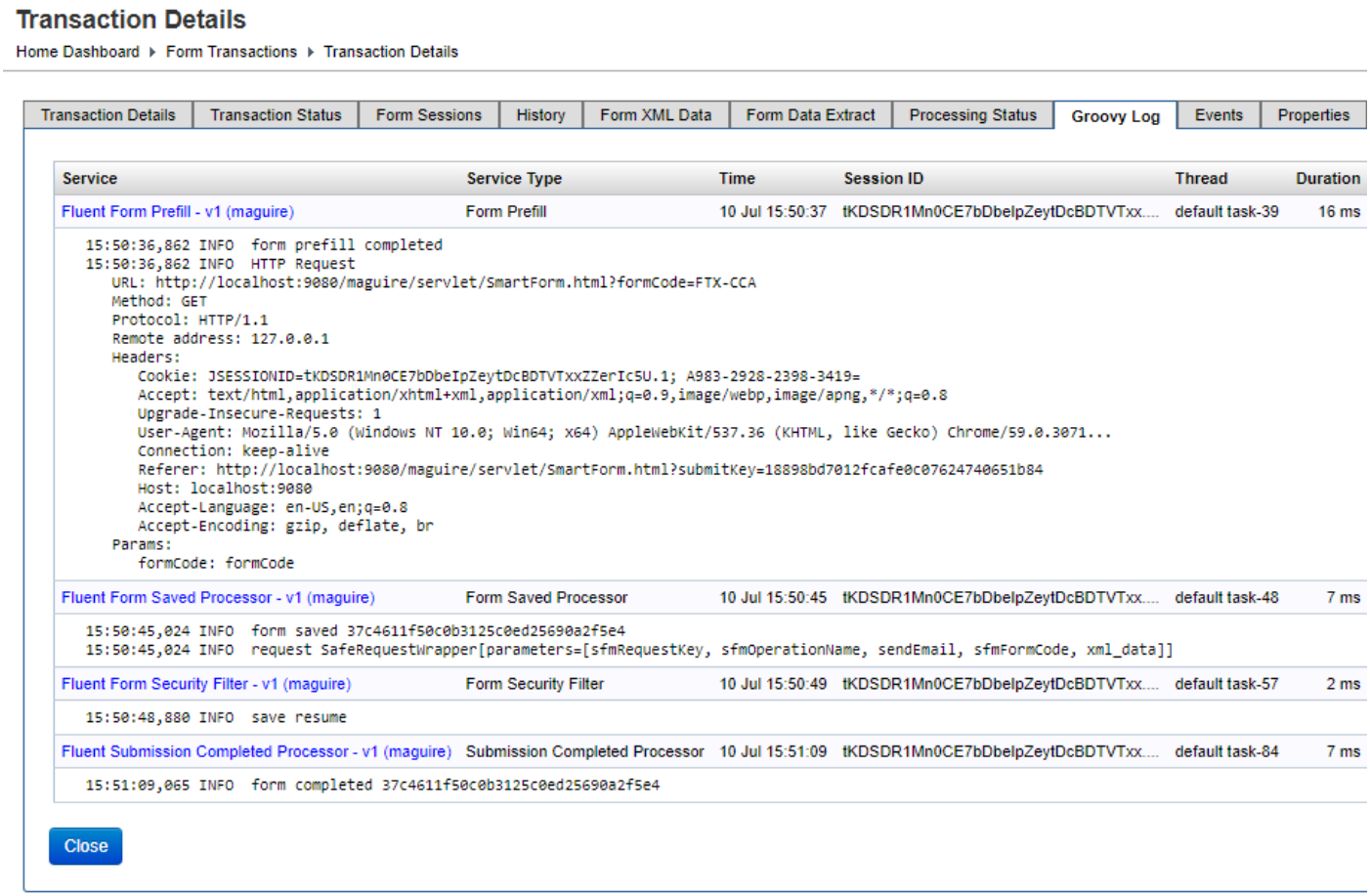
- http://www.groovy-lang.org/mailing-lists.html#nabble-td5722176

Service Logging

Transaction Manager provides a central Groovy Service Logging facility for operational support and system testing. This facility is extremely useful for monitoring and trouble shooting integration with remote services.

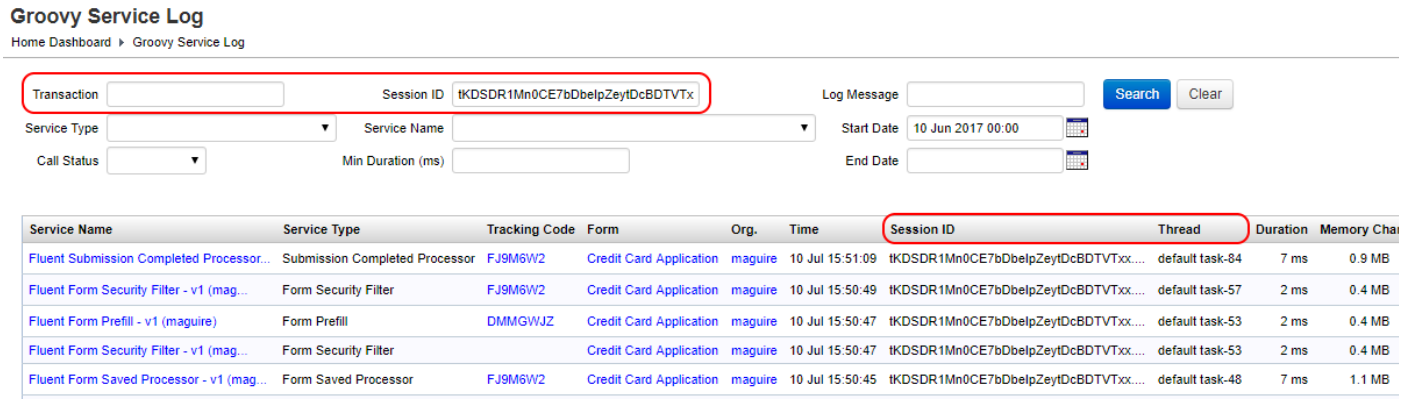
Transaction Details Groovy Logging

The Transaction Details operational support page includes a Groovy Log view which shows logging messages in-line, including Session and Thread information.



Groovy Service Log

The Groovy Service Log page can be used to search and view log entries. To use this navigate to Services > Groovy Service Log. A particularly useful search filter is the Session ID from the browser interacting with the form. This allows logging information related to all services consumed by a form in that session to be viewed together.



By default all Groovy Services invocations are logged. To disable Groovy Service Logging edit the Service Definition's service parameter 'groovyLoggingEnabled' and set this value to false.

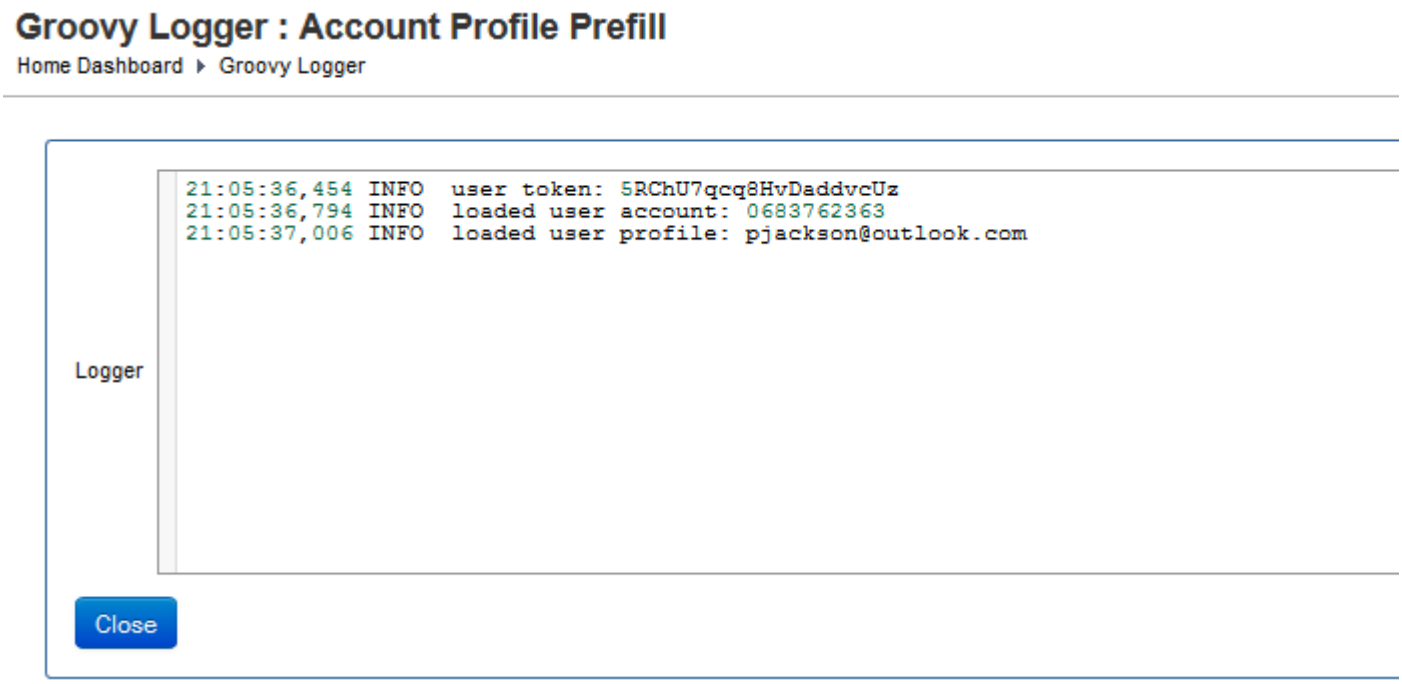
There is also a system level Deployment Property 'Groovy Logging Enabled' which specifies whether Groovy Service Logging is enabled if a Service Definition does not specify a 'groovyLoggingEnabled' parameter. This is relevant for Service Definitions created before TM version 4.2 which did not have this Service Parameter.

Groovy Logger

Transaction Manager also provides a **GroovyLogger** object which you can use to log messages from inside your Groovy scripts. This is extremely easy to do and you should use this when developing scripts and debugging operational issues. For example:

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.vo.*

import javax.servlet.http.*

class LoggingUserData {

    String invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {
        // Get user token cookie
        def userToken = ...
        logger.info 'user token:' + userToken

        // Load user account
        def account = ...
        logger.info 'loaded user account:' + account.id

        // Load user profile
        def profile = ...
        logger.info 'loaded user profile:' + profile.email
    }
}
```

When your Groovy service is run these logger messages will be associated with the Groovy Service Log record. In the Groovy Service Log you can click on the 'Groovy Logger' icon and see logger messages.



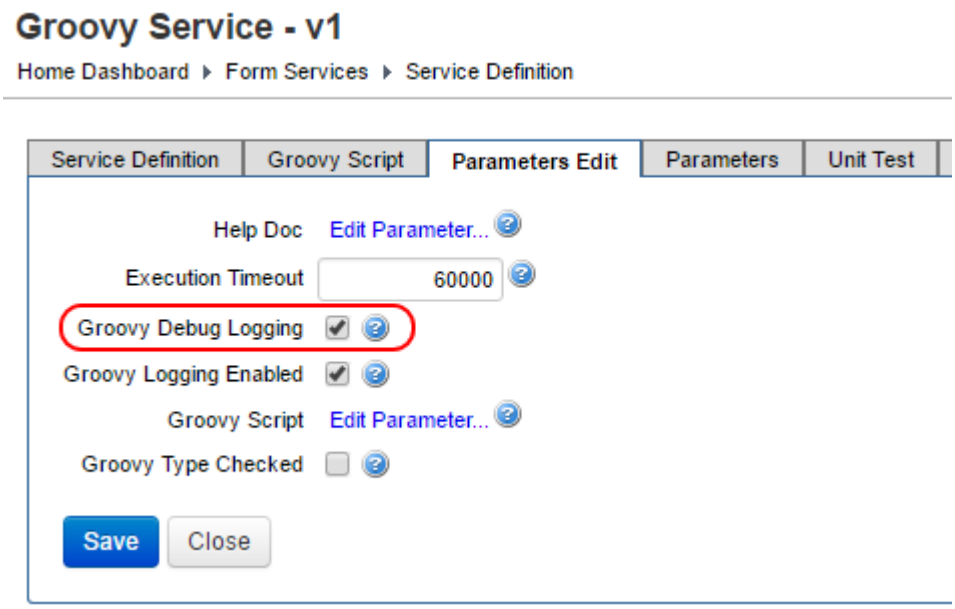
This allow you to view script execution logging messages without having to access the server log files. It is also much easier to read the logger messages of your call thread compared to reading server log files where there are many threads writing to the file as the same time.

Debug Logging

When a Groovy Service executes without any errors all INFO, WARN and ERROR level messages are logged, but DEBUG messages aren't. If the Groovy Service throws an exception however, all logging messages including DEBUG level are logged. The rational being when an error occurs you need all the information available to diagnose the issue, while during normal execution you only need to log INFO level messages.

Important Note: this is an extremely useful capability, put DEBUG logger message into your script as the points where you want to record info to help diagnose any operational issues.

If you need to override this default behavior and log all DEBUG level messages set the 'Groovy Debug Logging' service parameter to true.



This is a handy feature to diagnose operational issues, turn on debug logging to identify the issue, then turn if off once the issue has been resolved.

Server Log File

When Groovy scripts are being executed these logger statements are written to the server log file **`server/standalone/log/server.log`**. For example:

```
21:42:54,792 INFO  [com.avoka.fc.core.service.TransactionProcessor] (TM_Worker-1) Transaction Processor
completed in 0 sec.
21:43:35,312 ERROR [com.avoka.core.groovy.GroovyLogger] (pool-32-thread-1) java.lang.NumberFormatException: For
input string: "12,3"
21:47:54,795 INFO  [com.avoka.fc.core.service.TransactionProcessor] (TM_Worker-5) Transaction Processor
completed in 0 sec.
```

Note in the example above the **`logger.info`** statements were not logged to the server log file. This is because the Log4J GroovyLogger category logging level set to WARN which means WARN and ERROR messages will be logged, but DEBUG and INFO messages will not be.

You can change this configuration file: **`server/standalone/configuration/standalone.xml`**

```
<logger category="com.avoka.core.groovy.GroovyLogger">
    <level name="WARN"/>
</logger>
```

Logging Sensitive Data

Please note as these script execution log messages are stored in the database and in the server log file in clear text, do not log any information which is highly sensitive and should be encrypted. For example dont to this:

```
// DONT DO THIS:
logger.info "Form Data: " + submissionXml
```

If you do need to record additional sensitive data against a transaction please create a encrypted **`SubmissionProperty`** record and associate it with the transaction **`Submission`**.

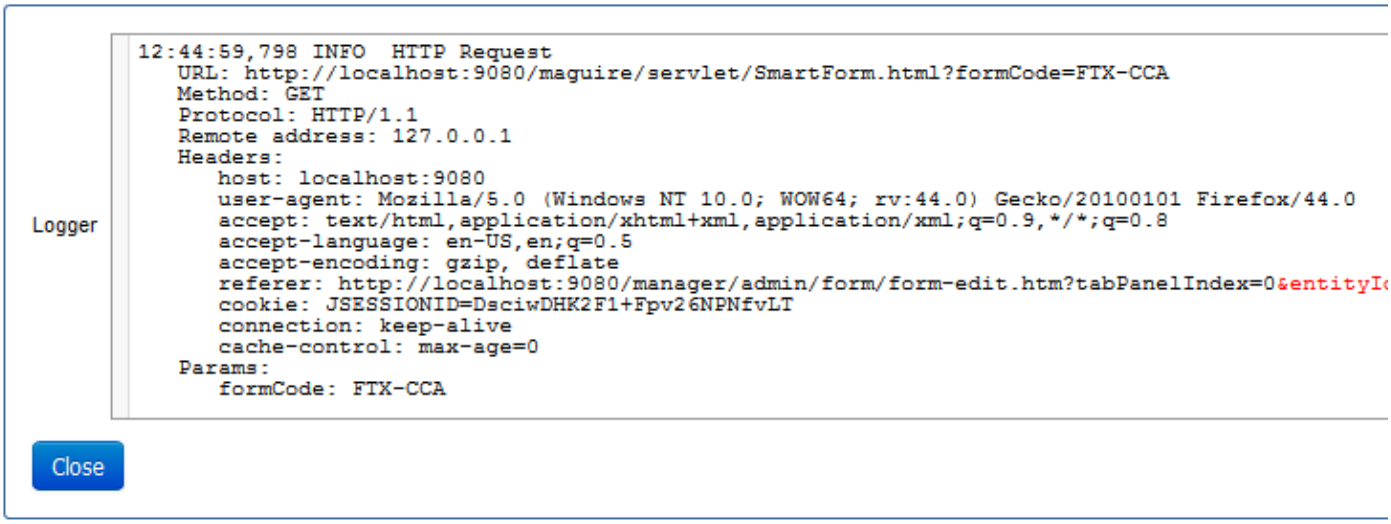
Logging HTTP Requests

The **`GroovyLogger`** provides special support for logging **`HttpServletRequest`** objects.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.vo.*

import javax.servlet.http.*

class LoggingRequestData {

    String invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {

        logger.info request

        //...
```

```
    }
}
```

The Groovy script above will log the service request output illustrated below.



3rd Party Java Libraries

Avoka Transact includes the following 3rd party Java libraries. Please note that Groovy services developed against the Transact Fluent SDK are restricted to access packages specified in Fluent SDK Security Configuration.

Library Name	Java Package / JBoss Module	Version
Apache Axis	org.apache.axis	1.4
Apache Camel	org.apache.camel.core	2.8.0
Apache Cayenne	org.apache.cayenne	3.1
Apache Click	org.apache.click	2.4
Apache Commons Command Line Interface	org.apache.commons.cli	1.2
Apache Commons Bean Utils	org.apache.commons.beanutils	1.8.0
Apache Commons Codec	org.apache.commons.codec	1.9
Apache Commons Collections	org.apache.commons.collections	3.2.2
Apache Commons Discovery	org.apache.commons.discovery	0.2
Apache Commons Email	org.apache.commons.email	1.3.1
Apache Commons FileUpload	org.apache.commons.fileupload	1.3.1
Apache Commons HttpClient	org.apache.commons.httpclient	3.0.1
Apache Commons IO	org.apache.commons.io	2.4
Apache Commons Lang	org.apache.commons.lang	2.4
Apache Commons Math	org.apache.commons.math	1.2
Apache Commons Net	org.apache.commons.net	3.3
Apache Commons Pool	org.apache.commons.pool	1.3
Apache CXF	org.apache.cxf	2.4.1
Apache FOP	org.apache.fop	1.0
Apache HTTP Components	org.apache.httpcomponents	4.5
Apache James Mime4J	org.apache.james.mime4j	0.6
Apache Log4J	org.apache.log4j	1.2.16
Apache Neethi	org.apache.neethi	3.0.0
Apache POI	org.apache.poi	3.10.1
Apache Santuario	org.apache.santuario.xmlsec	1.4.4
Apache Shiro	org.apache.shiro	1.2.1
Apache Velocity	org.apache.velocity	1.7
Apache WS Security	org.apache.ws.security	1.6.1
Apache WS XML Schema	org.apache.ws.xmlschema	2.0
Apache Xalan	org.apache.xalan	2.7.1
Apache Xerces	org.apache.xerces	2.9.1
Apache XMLBeans	org.apache.xmlbeans	2.6.0
Apache XML Resolver	org.apache.xml.xml-resolver	1.2
AWS SDK for Java	com.amazonaws	1.11.12
Bouncy Castle	org.bouncycastle	JDK15-154
ClamAVJ	com.philvarner.clamavj	0.1
FuseSource Common Management	org.fusesource.commonman	1.0
Google Gson	com.google.gson	2.2.4
Google Guava	com.google.guava	17.0
Groovy Lang	org.codehaus.groovy	2.4.7
Groovy WS-Lite	org.codehaus.groovy	0.8.0

Groovy HTTP Builder	org.codehaus.groovy	0.5.2
Hessian Remoting	com.caucho.hessian	4.0.37
iText PDF	com.itext	5.5.8
Java Cryptography Extension (JCE)	javax.crypto	-
Java Sysmon	com.jezhumble.javasysmon	0.3.3
Jaxen	org.jaxen	1.1.3
Jayway JsonPath	com.jayway.jsonpath	2.0.0
JBoss RESTEasy JAX-RS	org.jboss.resteasy.resteasy-jaxrs	2.3.5
Joda Time	org.joda.time	2.8.2
Jsch	com.jcraft.jsch	0.1.50
Json-Lib	net.sourceforge.json	2.4
JSoup	org.jsoup	1.8.3
Microsoft EWS Java API	com.microsoft.ews-java-api	2.0
Microsoft JDBC Driver	com.microsoft.sqlserver	sqljdbc 4
MVEL	org.mvel	2.2.6
MySQL JDBC Driver	com.mysql	5.1.39
PayPal Java SDK	com.paypal.sdk	76.0
QRGen	net.glxn.qrgen	1.4
Oracle JDBC Driver	com.oracle	11.2.0.3.0
Open SAML	org.opensaml	1.4.4
Quartz	org.quartz	1.8.6
Recaptcha	net.tanesha.recaptcha	0.0.8
SafeNet Luna	com.safenet.luna	5.3.0-11
SAXON	net.sourceforge.saxon	9.7.0-8
SLF4J	org.slf4j	1.6.1
Spring Framework	org.springframework.spring	3.1.4
Spring Security	org.springframework.security	3.1.4
Super CSV	org.supercsv	2.0.1
User-Agent-Utils	eu.bitwalker.useragentutils	1.20

Please note if you have classloader issues resolving some of these libraries you may need to contact Avoka support for assistance.

The Transact Project Template

The Transact Project Template is a pre-configured project structure containing Ant build files, features and conventions used when developing, deploying and testing Transact extension modules using a modern desktop IDE. It is recommended that all Groovy service development be performed using the framework provided by the Transact Project Template.

This project has a dependency on the Transact Fluent API and requires that you install the API project first. See:

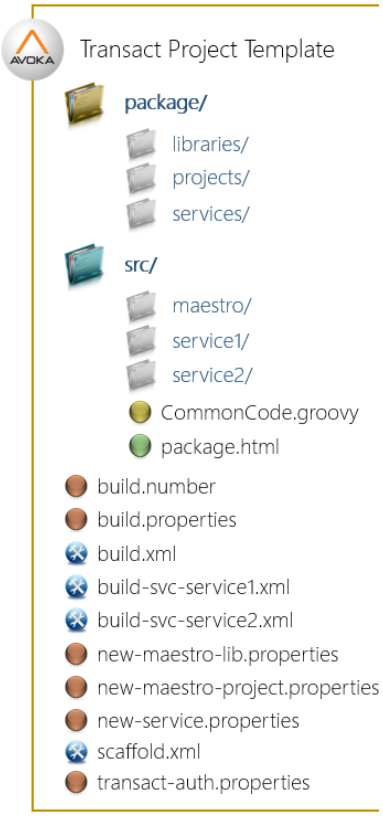
- Transact Fluent SDK Download for the latest versions of both projects
- Desktop IDE Development with The Transact Fluent API for instructions on configuring your IDE

In addition to support for Groovy service development, the Transact Project Template includes features relating to packaging and distribution of these services along with other asset types in a single archive file. Transact Distribution Packate (TPac) can contain any number of Groovy services as well as Maestro libraries, projects, and sample forms.

Project Structure

At a high level, the project structure includes:

- **package** folder containing the assets to be included in the distribution package:
  - libraries - containing Maestro library files with reusable UI widgets, templates, resources etc...
  - projects - containing Maestro projects with relevant design assets
  - services - containing generated service archives
- **src** folder containing all Fluent Groovy service source files:
  - a sub-directory will be created for each service generated
  - a sub-directory will be created to contain Maestro source files if required
  - common groovy classes to encapsulate code used by the services (optional)
  - package.html - used to document the high-level usage instructions for the package
- **build.number** file containing the current distribution build number
- **build.properties** file containing project level build configurations
- **build.xml** - the main project Ant build file with project level build targets (see Working with the Ant Build Files)
- **build-svc-<service.code>.xml** - a build file for each service in the project containing targets specific to a single service (see Working with the Ant Build Files)
- **new-maestro-lib.properties** file containing properties used in the Maestro library generation target (**create-maestro-lib**)
- **new-maestro-project.properties** file containing properties used in the Maestro project generation target (**create-maestro-project**)
- **new-service.properties** file containing properties used in the new service generation target (**create-new-service**)
- **scaffold.xml** - the seed Ant build file used to initialize the project and setup the files and directories
- **transact-auth.properties** file containing authentication details required to deploy assets into Transact Manager



See also:

- Transaction Distribution Package (TPac)
- Working with the Ant Build Files
- Unit Testing Groovy Services
- ServiceDoc Annotations

Transaction Distribution Package (TPac)

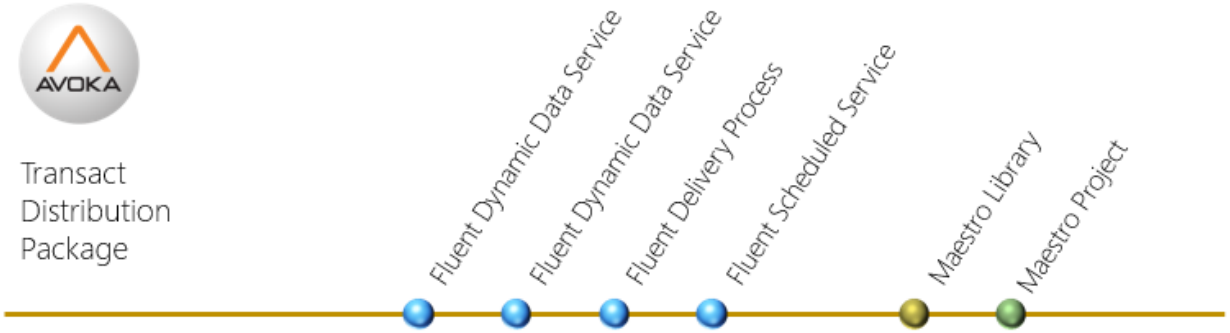
What is a TPac?

A TPac, short for 'Transact Distribution Package' is a ZIP archive containing related assets that extend the capabilities of the Avoka Transact platform. The contents of the TPac, structured in a manner that is recognized by Transact Manager, can include Groovy services, Maestro libraries / projects and documentation. TPacs are used to distribute these extension capabilities as a single archive file format.

The primary purpose of the Transact Project Template is to support developers as they create extension modules for the Transact platform. Groovy services developed against the Transact Fluent API are often a core element of these extension modules and the Transact Project Template aims to streamline Fluent Groovy service development. It is commonly a requirement to develop more than one service, each performing functions related to the same overall capability, so the Transact Project Template can grow to include as many services as required.

In addition to Groovy services, Transact extension modules commonly have a need for other asset types such as User Interface (UI) components that access the Groovy services. The development of these UI assets is done in Maestro (the Avoka design tool) and the reusable components get added to the design palette for use by designers as they use Maestro to develop forms. A Maestro library is the format in which these reusable assets are shared with users, while the Maestro project contains the original design assets. And occasionally there is a need to provide pre-built sample forms that demonstrate certain capabilities for knowledge distribution or training purposes and these may also be included in a Maestro project.

All of these assets together represent a set of related capabilities that are best grouped together and distributed as a single package. A **T**ransact Distribution **Pac**kage (**TPac**) is a simple archive file based on the ZIP format that supports the bundling of all these asset types.



The Transact Project Template has built-in support for creating these packages in a standard format, and generating help documentation describing the package content. These functions are facilitated by the Project Build File (see Working with the Ant Build Files for details).

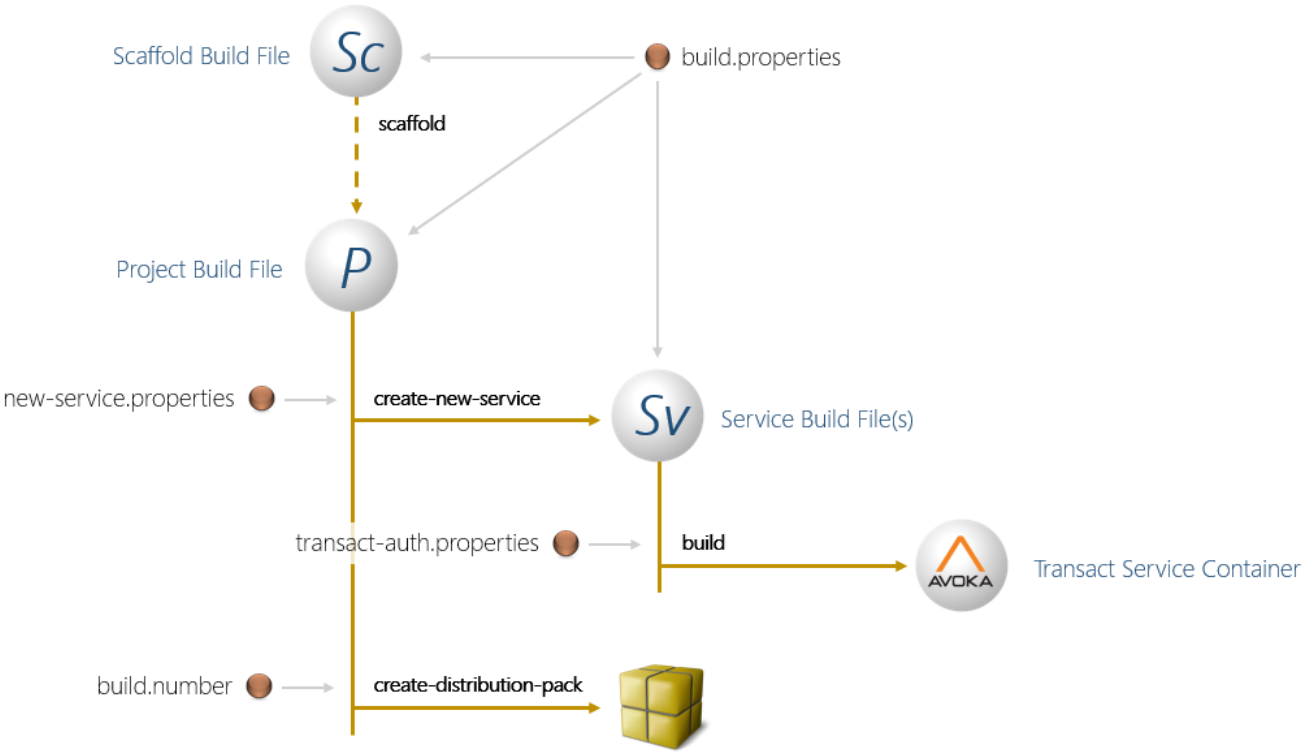
Working with the Ant Build Files

The Transact Project Template contains several types of Ant build file:

- The Scaffold Build File
- The Project Build File
- Service Build Files
- Maestro Build Files

The recommended usage pattern when working with these build files in a new project is as follows:

1. Before anything else, open and edit the values in the **build.properties** file as documented in that file. This properties file is used by all build files.
2. Run the **scaffold** target on the Scaffold Build File.
3. Edit the values in the **new-service.properties** file as documented in that file.
4. Run the **create-new-service** target on the Project Build File to generate the new service assets.
5. Edit the values in the **transact-auth-properties** file to contain credentials for the Transact Manager development environment.
6. *<Service development effort>*
7. Run the **build** target on the Service Build File to package, deploy and test the new service in Transact Manager.
8. *<Development Iteration>*
9. Run the **create-distribution-pack** target on the Project Build File to package the project assets into a single archive file.



The Scaffold Build file

The Scaffold Build File (named **scaffold.xml**) contains Ant targets that create all other required files in the project based on the version of the Fluent API you are using.

The primary build target (**scaffold**) is used to initialize the project ready for development. Execution of this target should be performed first and results in generated content based on templates provided in the **assets** folder of the **transact-fluent-api-x.x.x** distribution (@since 5.0.2).

The Scaffold build file contains the following targets:

Build Target	Description
scaffold	Initializes the project by creating the folder structure and files used by both the project and service build targets. This target should be run first, and there is usually no need to run it again unless you have corrupted your project or are upgrading to a new version of the sdk.
scaffold-<asset-name>	There are a bunch of sub targets that are executed by the primary scaffold target. These sub targets are responsible for the creation of individual elements of the project structure/content.
regenerate-all-svc-build-xml	Runs the **build** target of all service build files found in the project root directory for this project.

The Project Build File

The Project Build File (named **build.xml**) contains build targets that facilitate activities at the project level, including:

- New Service Generation: creation of new services based on the properties defined in the **new-service.properties** file.
- Build-All Functions: Execution of build targets contained in the service build files.
- TPac Distribution: creation of the Transact Distribution Package (TPac) containing the relevant project assets.

To achieve these functions, the Project Build File exposes the following targets:

Build Target	Description
build-all-services	Runs the **build** target of all service build files found in the project root directory.
package-all-services	Runs the **package** target of all the service build files found in the project root directory.
package-all-maestro-libs	Runs the **package-lib** target of all the Maestro library build files found in the project root directory.
package-all-maestro-projects	Runs the **package-project** target of all the Maestro project build files found in the project root directory.
create-new-service	Generates a new service based on the properties defined in the **new-service.properties** file and adds the source files to the **src** directory.
create-maestro-lib	Creates a new Maestro library based on the properties defined in the **new-maestro-lib.properties** file. Supports an option to download the new library from Maestro and adds the sources to the **src/maestro/lib** directory.
create-maestro-project	Creates a new Maestro project based on the properties defined in the **new-maestro-project.properties** file. Supports an option to download the new project from Maestro and adds the sources to the **src/maestro/project** directory.
create-distribution-pack	Creates a distribution package based on the contents of the **package** folder and places the package file in the **dist** folder. The name of the package file will contain the package major and minor version as found in the **build.properties** file and the build number as found in the **build.number** file. The build number is incremented by 1 each time this target is executed.
generate-package-docs (formerly *create-readme-doc*)	Runs the package documentation generator over the assets of the **package** folder to generate the help file (**readme.html**) and manifest file (**manifest.json**) to be included in the distribution archive. The contents of **src/package.html** are merged into the generated **readme.html** doc.
deploy-all-services	Runs the build target on all service build files
package-all	Packages all services, libs and projects

Service Build Files

A service build file is generated for every service you add to your project, so your project may have multiple service build files in the root project directory if you have created more than one service. Service build files are named **build-svc-<service.code>.xml** where the **service.code** is defined in the new-service.properties file prior to generation. The service code is also used for the naming of the service source directory in the **src** folder.

Use of service build files is recommended, but optional. The Project Build File contains aggregate targets that execute the same functions for all services but it is usually the case that you are working on a single service at any one time.

To create a new service you first need to complete the **new-service.properties** file before running the **create-new-service** target on the project build file.

Service build files contain the following targets:

Build Target	Description
build	Compiles, packages, uploads and tests the service against the remote Transact Service Container.
type-check	Performs the security checks and static type checking to identify any issues before deploying to the Transact Service Container
help-doc	Generates the service Help Doc according to the ServiceDoc Annotations found in the Fluent Groovy Script.
package	Performs the **type-check** and **doc** targets before creating the service archive and placing it in the package/services folder.
upload	Pushes the service archive up to the remote Transact Service Container reading from the **transact-auth.properties** file for authentication credentials.
deploy	Performs the **package** and **upload** targets in sequence for instances where you want to re-deploy a service after changes without running unit tests.
test-remote	Runs the Unit Test Script in the remote Transact Service Container reading from the **transact-auth.properties** file for authentication credentials.
undeploy (formerly *unload*)	Removes the service from the remote Transact Service Container reading from the **transact-auth.properties** file for authentication credentials.
validate-service-	Sub-target of the **doc** target to ensure the **service-def.json** file has certain properties set.

def	

Maestro Build Files

The Transact Fluent SDK supports synchronizing Maestro libraries and projects to your IDE. This allows you to store these assets in source control systems as you do with groovy services and also makes it possible to modify the source files and deploy your changes back to the server.

To add a maestro library to your IDE project you must first complete the **new-maestro-lib.properties** file before running the **create-maestro-lib** target on the project build file.

Similarly, to add a Maestro project to your IDE project you must first complete the **new-maestro-project.properties** file before running the **create-maestro-project** target on the project build file.

Each Maestro asset you synchronize to your IDE will have its own build file and like the service build files, you may have more than one in your project. Maestro build files are named as follows:

- **build-lib-<library-code>.xml** where **library-code** is generated by the system based on the library name.
- **build-project-<project-code>.xml** where **project-code** is generated by the system based on the project name.

Maestro assets may be created from scratch, but more commonly they will be seeded from existing assets in the Maestro environment. The properties files contain attributes to control this behavior:

- new-maestro-lib.properties :: **download.existing.library**
- new-maestro-project.properties :: **download.existing.project**

If these properties are set to **true**, the create target will search for an asset on the Maestro server with the same name and if found will use it as a seed.

Maestro library build files contain the following targets:

Build Target	Description
deploy-lib	Runs the **package-lib** target then deploys the library to the Maestro server, reading from the **transact-auth.properties** file for authentication credentials. Note: if the library already exists on the server it will be overwritten and no backup will be kept.
download-lib	Synchronizes all assets held in the Maestro library on the Maestro server down to the IDE and unpacks the library in the relevant **src/maestro/lib** directory, reading from the **transact-auth.properties** file for authentication credentials.
package-lib	Packages up the library assets from the relevant **src/maestro/lib** directory and stores the zip archive locally in the **/tmp** folder. This target evaluates the **distribute.library** property defined in the library build file and if true the library archive will be copied to the **/package/libraries** folder so that it is included in the distribution.

Maestro project build files contain the following targets:

Build Target	Description
deploy-project	Runs the **package-project** target then deploys the project to the Maestro server, reading from the **transact-auth.properties** file for authentication credentials. If the project Note: if the project already exists on the server, a backup will be made of the existing project and stored in the **/tmp** folder locally.
download-project	Synchronizes all assets held in the Maestro project on the Maestro server down to the IDE and unpacks the project in the relevant **src/maestro/project** directory, reading from the **transact-auth.properties** file for authentication credentials.
package-project	Packages up the project assets from the relevant **src/maestro/project** directory and stores the zip archive locally in the **/tmp** folder. This target evaluates the **distribute.project** property defined in the project build file and if true the project archive will be copied to the **/package/projects** folder so that it is included in the distribution.

Unit Testing Groovy Services

To develop your services business logic you should write unit test code to execute your Groovy script under a variety of test scenarios. If you develop your services in this way you will be more productive and can develop much more robust code. Unit Tests enable you to exercise Groovy services in ways which are really difficult to produce manually.

Most Groovy service templates provide you with a working Groovy Script and Unit Test to get started with.

When developing Unit Tests use the **`assert`** keyword to ensure your code is producing the correct results.

The Hello World Service Example shows how you can write and execute unit tests using the Transact Fluent SDK.

Unit Testing Execution

You can code and execute unit tests directly in the IDE where you have the Fluent SDK installed, or inside the Transact Manager console.

Your IDE will provide additional features to make unit test code easier to develop and manage, such as:

- Code completion
- Syntax highlighting
- Error highlight and hints
- Context help
- Snippets
- Source control integration

The Fluent SDK will also deploy your unit tests to Transact Manager.

Supported libraries for Unit Testing

There are several supporting frameworks and libraries that you can take advantage of in your unit test code, including:

- JUnit 4 - a popular unit test development and execution framework
- Mockito - a popular mocking library
- Mock VO Constructors - Avoka-supplied mock classes
- Mock Register - Avoka-supplied mock environment, particularly useful for mocking HTTP responses

JUnit4 Support

Support for JUnit4 test annotations enables developers to write service unit tests which can be executed both locally or remotely on a TM server.

The JUnit4 `@Test` annotation can be used specify multiple test methods to be executed in a groovy unit test.

An example unit test with multiple methods is provided below. Please note the `AbstractJUnitTest` class provides public `svcDef` and `testParams` attributes for use in the test code which are populated when executed on a TM server.

```
import com.avoka.tm.test.*
import org.junit.Test

class UnitTest extends AbstractJUnitTest {

    @Test
    void test1() throws Exception {
        // First test method
    }

    @Test
    void test2() throws Exception {
        // Second test method
    }
}
```

Other supported JUnit4 annotations include:

```
    @BeforeClass - specifies method to be called once when test class in setup
    @Before      - specifies method to be called before each test method invocation
    @Test        - specifies test method to be invoked
    @After       - specified method to be called after each test method invocation
    @AfterClass  - specifies method to be called once when test class is finished
    @Ignore      - specifies test method that will not be executed
```

Mockito Support

Support is provided for the Mockito http://site.mockito.org/ unit testing mocking framework. The Mockito library is included in the Fluent SDK distribution and has been added to the Secure Compiler Whitelist for unit test execution. Please note that the Mockito library is not white listed for production execution as if provides extensive reflection and interception methods which could compromise the Fluent services security model.

Mock VO Constructors

The Fluent Value Object (VO) classes provides immutable value objects for core TM entities represented in the database. You can create these mock objects using the VO constructors which take a map of field attributes. This enables you to mock VO classes without having to execute your unit tests in a TM server.

An example mock Txn object is provided below:

```
import com.avoka.tm.vo.*

// Unit Test Code
Map fields = [
   "id": new Long(123),
   "formCode": "CCA-FTX",
   "userSaved": true
]

Txn tx = new Txm(fields)
...
```

Please note for security reasons these VO map constructors are not permitted in production code execution.

Mock Register Support

Support is provided for developers creating Groovy services calling HTTP endpoints to mock HttpResponse values. This can be very useful when the 3rd party service isn't available yet, or you need to test error scenarios which the HTTP endpoint does not normally provide.

An example is provided below:

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

class UnitTest extends AbstractJUnitTest {

    @Test
    void test() throws Exception {
        Map svcParams = [
            "name": "Form Validator",
            "versionNumber": 2,
            "clientCode": svcDef.clientCode
        ]
                SvcDef formSD = new SvcDef(svcParams);

        new MockRegister().when(formSD).thenReturn("{ 'status': 'OK' }");

        // Call service
        Map params = [
            "svcDef": svcDef
        ]

        String result = (String) new ServiceInvoker(svcDef).invoke(params)

        // Test Results
        ...
    }
}
```

The TestMockRegister also supports mocking responses when calling other Groovy Services from your service code.

In the example below our service unit test is invoking another service called "Form Validator" and having this service return the JSON "{ 'status': 'OK' }".

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

class UnitTest extends AbstractJUnitTest {

    @Test
    void test() throws Exception {
        Map svcParams = [
            "name": "Form Validator",
```

```
            "versionNumber": 2,
            "clientCode": svcDef.clientCode
        ]
                SvcDef formSD = new SvcDef(svcParams);

        new MockRegister().when(formSD).thenReturn("{ 'status': 'OK' }");

        // Call service
        Map params = [
            "svcDef": svcDef
        ]

        String result = (String) new ServiceInvoker(svcDef).invoke(params)

        // Test Results
        ...
    }
}
```

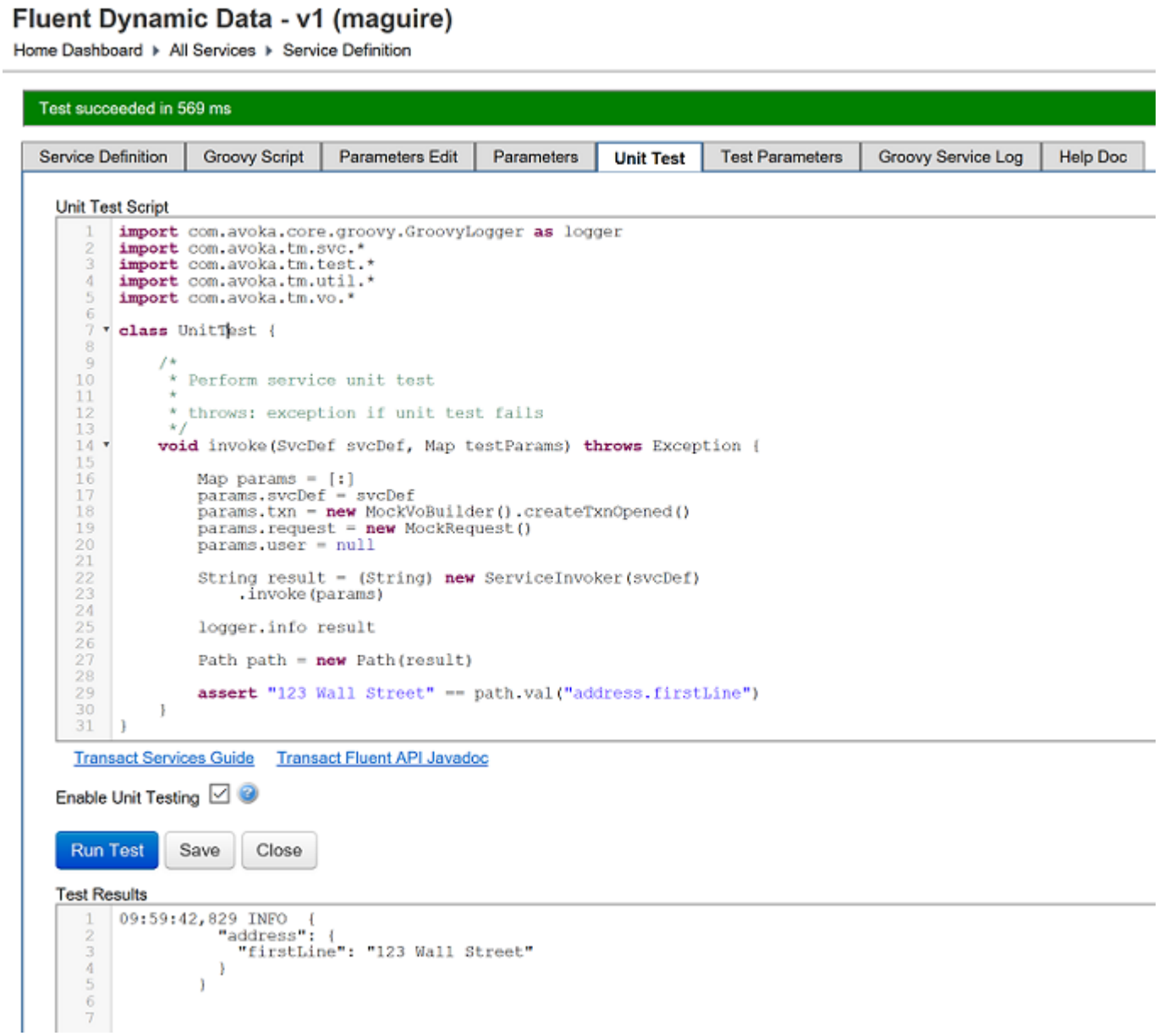
Executing Unit Tests in your IDE

You can choose any IDE that supports Ant tasks, including IntelliJ and Eclipse. The methods you need to run unit tests in your particular IDE will differ according to your chosen IDE.

You can find installation and usage instructions for some IDEs at Desktop IDE Development with The Transact Fluent API

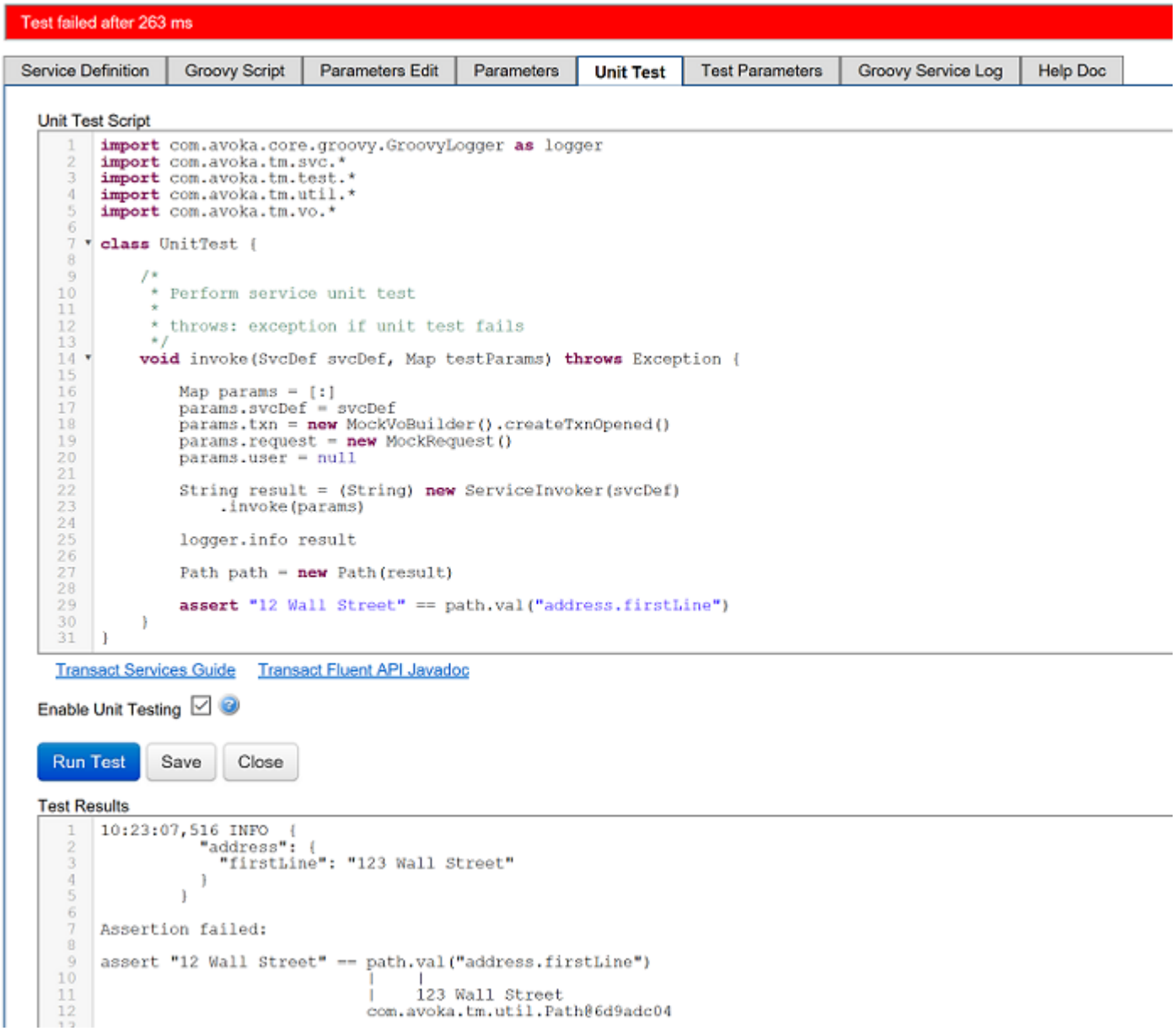
Executing Unit Tests in Transact Manager

Groovy service unit tests can also be executed directly in the Transact Manager console by accessing the service and selecting the Unit Test tab.



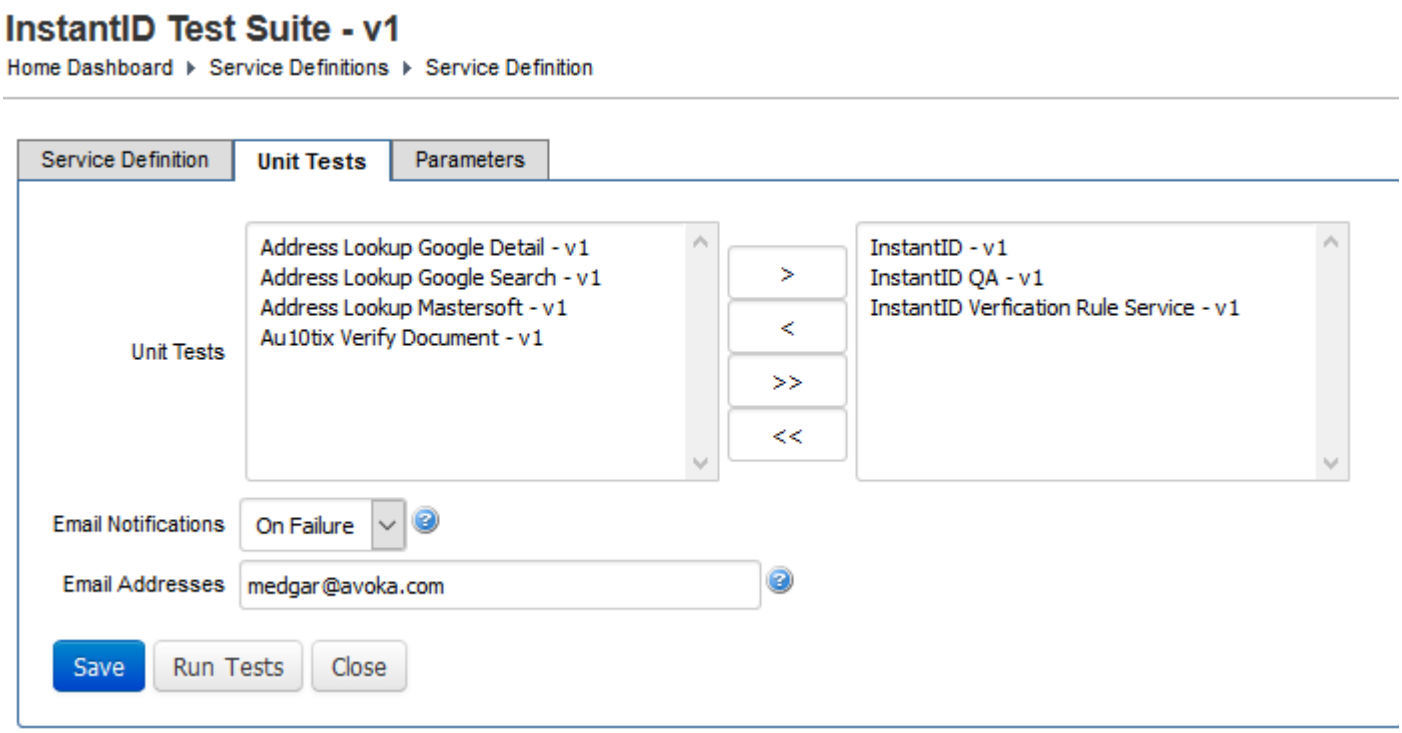
In the 'Unit Test' tab click on the 'Enable Unit Testing' option and then click on 'Run Test'. If the Unit Test ran successfully and did not throw any exceptions you will get a Green bar.

If the Unit Test throws an exception you will get a Red bar indicating a failure.



Service Test Suites

Use Service Test Suites to enable Continuous Integration (CI) testing of Groovy Services. These test suites can be setup to run the unit tests on a group of services with regular scheduled execution and notification email reports.


InstantID Test Suite - v1
Home Dashboard ▸ Service Definitions ▸ Service Definition
Service Definition
Unit Tests
Parameters
Unit Tests
Address Lookup Google Detail - v1
Address Lookup Google Search - v1
Address Lookup Mastersoft - v1
Au10tix Verify Document - v1
>
<
>>
<<
InstantID - v1
InstantID QA - v1
InstantID Verfication Rule Service - v1
Email Notifications
On Failure
Email Addresses
medgar@avoka.com
Save
Run Tests
Close

ServiceDoc Annotations

Services deployed into Transact Manager support the inclusion of an HTML help file containing usage instructions for consumers of the service. Using ServiceDoc you are able to automatically generate this documentation by annotating your groovy code. This has several key benefits:

- Creates consistency in the structure and content of help documentation across the board as document templates are used to create identically structured content for all services
- Removes the need to document the service independent of the groovy script, saving the developer a bunch of time stuffing around with HTML content and formatting, and duplicating content between the groovy code comments and the help documentation
- Improves the accuracy of help documentation as it is automatically updated before updates are deployed to the server - not all developers can be relied upon to manually update the documentation

ServiceDoc generates the service-help.hml file based on a template that includes content such as compatibility details, input/output parameters, 3rd party endpoint attributes, and sample code. ServiceDoc also parses the **service-def.json** file to extract information including service connection name and service parameters so ensure you use the **description** attribute to document your service parameters appropriately in **service-def.json**.

By default, the Transact Project Template is configured to generate the service help doc in this manner. This article describes how to document your services using this feature.

Wrap all ServiceDoc Content in a Comment Block

ServiceDoc ignores all content of your groovy script unless it is enclosed in a comment block starting with either /** or /* and ending with */. Single line comments using the // notation are also ignored.

```
        /* This is a single line block comment and is reviewed by ServiceDoc */

        /*
         * This is a multi-line block comment, also checked by ServiceDoc
         */

        // This is a line comment and is ignored by Service Doc
```

Service Description as the First Un-Annotated Block

General descriptive content about the service is generated from the first comment block before any annotations are encountered. This block is typically located directly above the class declaration. In the example below, everything down to the @since annotation is read as the service description to appear at the top of the help documentation:

```
/**
 Service description goes here...

 Any content up until the first annotation or end of comment block will be included in the documentation of the
service description.

 The content may include HTML tags like <b>bold</b> and structure like:
 <ul>
  <li>Item one</li>
  <li>Item two</li>
 </ul>

 @since Manager 5.0

 */
class MyGroovyClass {
```

Supported Annotations

The following table lists the supported ServiceDoc annotations that are interpreted to generate the help documentation:

Annotation Key	Usage	Example
@since	Used when documenting minimum supported versions of requried modules.	@since Manager 5.0
@in	Used to document an input parameter, provided in the request.	@in countryCode {O} Optionally specify a country code to limit search scope. Expects ISO standard 2 character codes (e.g. AU, US)
@out	Used to document an output parameter, returned in the service response.	@out verifyResult The outcome of the verification check - value will either be VERIFIED or UNVERIFIED
@svcConn	Used to document the required attributes of associated Service Connections	@svcConn Endpoint The URL to the 3rd party - defaults to https://verify.3rdparty.com/services

@sample	Used do document sample request / responses	@sample Request <pre> { "searchString": "1a Rialto Ln, Manly", "format": "long", "includePostal": true } </pre>

Annotation Structure

All annotated content is documented using a similar structure where the string immediately following the annotation is read as the attribute name, and all other content assumed as attribute description:

```
<annotation-key> <name> [<modifiers>] <description>
```

where:

- **<annotation-key>**: one of the supported annotation keys listed above.
- **<name>**: A single word string containing no white space (e.g. countryCode), or a multi-word phrase enclosed in single/double quotes (e.g. "Country Code").
- **<modifiers>**: Optionally include special tokens that apply additional attributes to the annotation such as {R} for Required or {O} for optional.
- **<description>**: All content on the same or following lines until the comment block is closed or next annotation is encountered.

Some examples follow:

Annotation	Name	Modifiers	Description
@since Manager 5.0	Manager		5.0
@in maxResultSize {O} Optionally specify a maximum result size	maxResultSize	{O} = Optional	Optionally specify a maximum result size
@svcConn Username {R} The login name used to authenticate with the 3rd party service	Username	{R} = Required	The login name used to authenticate with the 3rd party service
@sample "Error Response" <pre> { "errorCode": "SYSTEM_ERROR" } </pre>	Error Response		<pre> { "errorCode": "SYSTEM_ERROR" } </pre>

Example Script

The following example script demonstrates usage of annotations to document the service:

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.vo.SvcDef
import com.avoka.tm.vo.Txn
import com.avoka.tm.vo.User
import javax.servlet.http.HttpServletRequest
import groovy.json.JsonBuilder

/**
 This Dynamic Data service provides real-time searches on the Australian Business Register and can accommodates
searches via ABN, ACN or Business Name.

 @since Manager 5.0

 @svcConn Endpoint {R} Should point to the Super Fund Lookup REST API at: http://abr.business.gov.au/abrxmlsearch
/AbrXmlSearch.asmx
 @svcConn "Auth Key" {R} Should contain the GUID value for the client organisation. Each client must complete the
  <a href="http://abr.business.gov.au/Documentation/UserGuideWebRegistration.aspx">free registration process</a>
  in order to access the ABN Lookup web service.

 @sample Request
  <pre>{ "searchString": "47099389509" }</pre>

 @sample "Successful Response"
  <pre>{<br>
  "executionStatus": "SUCCESS",<br>
  "organisationName": "AVOKA TECHNOLOGIES PTY LTD"<br>
  "abn": "47099389509",<br>
  "acn": "099389509",<br>
  "status": "Active",<br>
  "entityTypeCode": "PRV",<br>
  "entityType": "Australian Private Company",<br>
  "addressState": "NSW",<br>
  "addressPostcode": "2095"<br>
  }</pre>
```

```
 @sample "Data Error Response"
  Provided when the an error with the input data was identified while handling the request:<p>
  <pre>{<br>
  "executionStatus": "DATA_ERROR",<br>
  "error": "Search text is not a valid ABN or ACN: xxxxx"<br>
  }</pre>

 @sample "System Error Response"
  Provided when the an anrecoverable system fault was identified while handling the request:<p>
  <pre>{<br>
  "executionStatus": "SYSTEM_ERROR",<br>
  }</pre><br>
  Note that no additional details are provided to the for this type of error.

 */
class AbrAbnSearch {

    String invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {

        // Check we have all the service details
        if(!svcDef.svcConn){
            throw new RuntimeException('Service Fault: No service connection configured')
        }
        if(!svcDef.svcConn.endpoint){
            throw new RuntimeException('Service Fault: No service connection endpoint configured')
        }
        if(!svcDef.svcConn.param1){
            throw new RuntimeException('Service Fault: No service connection auth key configured')
        }

        BusinessLookupAPI api = new BusinessLookupAPI()
                .txn(txn)
                .svcConn(svcDef.svcConn)

        /* @in searchString {R} The ABN (Australian Business Number) or ACN (Australian Company Number) of the
target organisation */
        String searchString = request.getParameter('searchString')
        logger.debug('Lookup ABN/ACN provided: ' + searchString)

        DynamicDataResult result = api.executeAbnAcnLookup(searchString)

        /*
         * @out organisationName The name of the organisation
         * @out abn The ABN (Australian Business Number) associated with the organisation
         * @out acn The ACN (Australian Company Number) associated with the organisation
         * @out status The current status of the organisation
         * @out entityTypeCode The short 3 character code identifying the organisation type e.g. PRV (see
         *       <a href="http://abr.business.gov.au/Documentation/UserGuideReferenceData.aspx">Reference Data<
/a>
         *       for a full list of codes and descriptions)
         * @out entityType The descriptive name for the organisation type e.g. Australian Private Company
         * @out addressState<br>addressPostcode
         *       The state and postcode that the organisation is registered in
         * @out executionStatus The status of the service execution [ SUCCESS | DATA_ERROR | SYSTEM_ERROR ].
         *      Successful execution will be denoted by a SUCCESS value. DATA_ERROR will indicate that there
         *      was an issue identified with the input data that may be resolved and potentially retried by the
user.
         *      SYSTEM_ERROR indicates that there was an unrecoverable system fault and the form should fall-back
         *      gracefully to an alternative path.
         * @out errorMessage In the event of a DATA_ERROR, this value may contain more information about the
error.
         */
        String abrResult = new JsonBuilder(result).toPrettyString()
        logger.debug 'Result\n' + abrResult
        return abrResult
    }
}
```

Coding Conventions & Practices

The Groovy coding conventions and practices referenced below are intended to promote consistency and reliability in the services built for the Avoka Transact platform. It is recommended that anyone developing Transact services, for what ever purpose, review and adhere to these guidelines.

Any TPAC being proposed for inclusion in the Avoka Exchange program will be assessed according to its compliance with these guidelines before being accepted into the program.

Using Defined Type Declarations in Favor of Dynamic Types

One of the conveniences that Groovy provides is dynamic type assignment. For example, by specifying a variable to be of type '**def**' the developer can avoid having to give that variable a definite type and can handle different types at run time.

This is similar (but not exactly the same) as specifying a variable to be of type **Object** in Java. For example, the following Groovy code compiles without needing to cast the name variable to a **String** type:

Groovy Dynamic Type Example

```
def name = "Stephanie"
println name.toUpperCase() // no cast required
```

However, in Java you need to explicitly cast the name variable to a **String** before using a method of the **String** class:

Java Object Example

```
Object name = "Stephanie";
System.out.println(((String) name).toUpperCase());
```

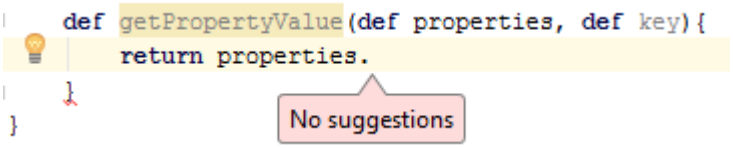
However, while this feature may give the coder an immediate convenience, there are a number of key reasons why it is better to use defined types:

- Make better use of IDE productivity assistants
- Reduce the chances of bugs
- Improve the readability of code

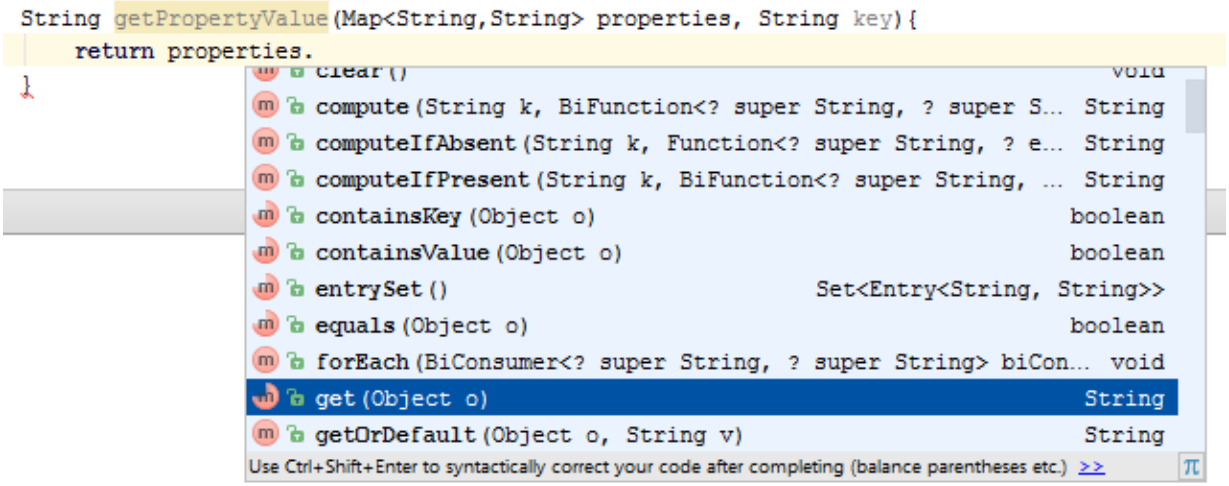
Activate IDE Productivity Assistants

IDE's are able to provide better code complete functions and type checking where defined types are used, making development and maintenance of Groovy code far more productive.

In the method declaration below we're expecting the first parameter to be a Map but the IDE does not know that and cannot provide any suggestions as to what functions and attributes are available on that object:



Using defined types activates the IDE productivity assistants:



Reduce the Chances of Run-time Bugs

Using defined types allows the IDE and compilers to identify bugs early so that they can be resolved before the software is deployed and running in a production scenario.

This improves the reliability of the code and reduces the operational burden of dealing with issues in live environments.

Improve the Readability of Code

Defined type declarations make code easier to read and to understand the intention of the code, and actually helps self-document the code.

Consumers of the code will be better informed as to the supported uses of the functions and less inclined to develop unsupported code.

Developers tasked with maintaining the code will have a greater understanding of the intended purpose of the code and less inclined to modify the functions to include unintended behaviors.

When does it make sense to use *def?*

There are some scenarios where it does make sense to use dynamic types, just don't let laziness be the reason to use **def**.

Use **def** only if there's no definite type applicable to the variable at compile time.

Naming Conventions for Groovy Services

When developing code of any type it is beneficial to apply naming conventions to improve the consistency and maintainability of the software. The suggested naming conventions in this article have been developed by Avoka based on our experience of what works best for Transact Groovy services.

Service Naming

Service names should be succinct and consist of plain English that clearly describes the function being performed. Services that integrate external systems should be named using the vendor/product name followed by the function being performed with a hyphen in between. Examples:

- greenID - Identity Verification
- FIS - Password Expiry Check
- FIS QualiFile - Verification
- ATO Super Fund Lookup - ABN Search
- ATO Super Fund Lookup - Quick Search

When naming your groovy services and associated service connections the words 'Service' and 'Connection' are redundant so omit these from the name. Similarly there is no need to include the type of service in the service name (e.g. 'DDS' for Dynamic Data Services)

Parameter Naming

When naming service parameters, request parameters, local variables and JSON data elements, you should always use camel case with a lower case first letter and capitals for the first letter of subsequent words (no hyphens, underscores or other special characters). E.g.

- clientName
- gender
- businessName
- driversLicenseNumber
- serviceResponse

Acronyms (with all capitals) should be treated as if they were a normal word, E.g.

- ufoLandingSite
- applicantDob
- licensedOrByo
- nameOfCeo
- favoriteNflTeam

Submission Property Naming

When writing submission properties relating to 3rd party service connectors, the property name should be prefixed by the service name using dot notation <service-name>.<property-name> E.g.:

- greenID.verifyStatus
- QualiFile.transactionTrackingId

Writing Null-Safe Groovy Code

Using some simple techniques to write null-safe code avoids many common causes of run time errors due to NullPointerExceptions.

See the Groovy documentation for detailed listing of operators:

- http://docs.groovy-lang.org/latest/html/documentation/index.html#groovy-operators

Null-Safe Navigation Operator ?.

The Null-Safe Navigation operator is used to avoid NullPointerExceptions where the existence of a valid object at run time is not guaranteed. Typically when you have a reference to an object you might need to verify that it is not null before accessing methods or properties of the object. Using if statements to perform this check when navigating deep into object graphs quickly becomes a lengthy and annoying coding exercise:

```
// Get the firstName request parameter and trim any leading or trailing white space
firstName = request.getParameter('title')
if(firstName != null){
        firstName = firstName.trim()
}
```

Using the Safe Navigation operator, these if statements can be avoided and a more concise null safe coding style can result:

```
firstName = request.getParameter('title')?.trim()
```

In the statement above, if the request parameter is null, the trim() function will not be executed and a NullPointerException will be avoided.

The Elvis Operator ?:

The Elvis operator is a shortening of the ternary operator and is very convenient for assigning to a variable a sensible default value where it would otherwise be set to null. In the code examples below the ternary operator requires you to repeat the value you want to assign if the value tested is not null or false, where the elvis operator will use the value tested if it is not null or false.

```
// Ternary operator
firstName = request.getParameter('title') ? request.getParameter('title') : ''
// Elvis operator
firstName = request.getParameter('title') ?: ''
```

Groovy Error Handling & Reporting

There are 2 main types of errors that occur in groovy service execution:

1. **Data Error**
   An issue was identified with some of the data provided to the service. This is usually a recoverable error that can be handled by the groovy service or, in the case of dynamic data services, reported to the user such that they may resolve the issue and try again. Examples may be that not enough information was provided or there was a format validation error with some of the data (e.g. invalid Social Security Number). It is generally best if these errors are intercepted before the service is called (e.g. in the form validation logic) to avoid these service level data errors.

2. **System Error**
   An unrecoverable error was received and the execution of the service failed. This type of error is commonly encountered when performing integrations with 3rd party services. It is important that these errors are logged to the System Error log or Event Log such that the Transact system monitoring service will pick them up and alerts generated.

Handling of these errors in groovy code should usually involve including error type information in the service response such that the consumer may handle the situation appropriately. For system errors, it is best not to provide any detailed messaging in the response as this may create security concerns. For data errors, some additional information about the nature of the error is often useful. E.g.

Data Error

```
{
    "verifyStatus": "UNVERIFIED",
    "executionStatus": "DATA_ERROR",
    "errorMessage": "Value of [someone@somewhere] for email is invalid"
}
```

System Error

```
{
    "verifyStatus": "UNVERIFIED",
    "executionStatus": "SYSTEM_ERROR",
}
```

Logging Third Party Service Calls

When utilizing third party services in groovy service definitions it is important to record each successful usage. Usage statistics are very useful and sometime required to be tracked for reconciliation and billing purposes.

The TxnUpdater provides a function to record these events as demonstrated below.

The service call should be logged only if:

1. A successful response has been received from the service - this can be verified with the convenience function **isSuccess()** on the HttpResponse object.
2. No data errors were reported in the response from the service. Typically a 400 (BAD REQUEST) response is received when there is an issue with the data provided in the request, however some 3rd integration APIs will incorrectly return a 200 response type but still report data errors in the body of the response (e.g. 'Invalid SSN') - these will need identified and the service log skipped in this scenario as data errors do not typically represent a billable event.

For billing purposes this is important because unsuccessful calls should not be billed. See the following code sample for an example usage scenario:

Sample Code

```
import com.avoka.tm.svc.TxnUpdater
import com.avoka.tm.http.GetRequest
import com.avoka.tm.http.HttpResponse
...
        try {
            response = new GetRequest(serviceEndpoint).execute()
        } catch(Exception e){
            // Log and return
                new EventLogger().setTxn(txn).setMessage(e.getMessage()).logError()
                return result.systemError()
        }

        if (response?.isDataError() || checkBodyError(response)) {
                return result.dataError(getDataErrorMsg(response))
        }

        if (response?.isSuccess()) {

                // Add the 3rd Party Service Call Log entry
                new TxnUpdater(txn).addServiceCallLog(serviceName, 'More Info (optional)', serviceEndpoint).
update()

                // Now process the response
                ...
        }
```

The inputs to this addServiceLog function are:

- **svcName**: The name of the third party service - this is not the SOAP Action name but rather relevant name that clearly identifies the vendor and service being called (e.g. FIS - QualiFile)
- **info**: Any additional details you may want to provide to assist in auditing, this may include key attributes included in the call (optional)
- **url**: Where the third party provides multiple endpoint URLs for different environments it is important to record this URL as calls to the test environment would be excluded from billing transactions. See also Managing multiple service endpoints and credentials for external service calls.

Note: remember you must call the **update()** funtion on the TxnUpdater to commit the log record

Encapsulating Common Code in Shared Groovy Classes

It is often the case that a suite of related groovy services will have some element of common requirement. Rather than duplicating the required functions in each of your services, you should instead create additional Groovy classes that may be shared among the services.

You can utilize common groovy classes in your services either by:

1. Add them to your Fluent SDK project so that there are no external dependencies; or
2. Keep them separate so they can be referenced by multiple Fluent SDK projects.

The process is very similar for these 2 options:

1. Create the common Groovy classes as required.
    a. Ensure that your classes are in the default package (i.e. they do not have a package declaration). Groovy classes in a package other than the default will not be permitted by the Fluent SDK Security Configuration.
    b. If you are adding these classes to your Fluent SDK project, add the source files to your sources root folder.
    c. If you are referring to these files externally you will need to ensure that the relevant class files are on the project build path for your IDE to utilize code complete and avoid class resolution errors.
2. Add the path of the common Groovy source file to your **service-def.json** file.
    a. These files must be added to the **fileIncludes** attribute of the **groovyScript** parameter
    b. The path must be a relative path from the service directory to the source file (e.g. "../MyCommonClass.groovy")

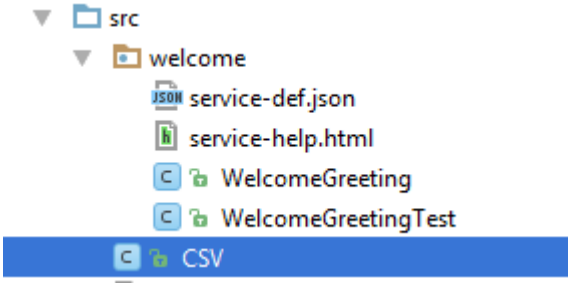
Common Code Example

In this example we will modify the Hello World Service Example to extrapolate the code that deals with CSV values into a common class that can be reused by other services in the project and has additional features and fail-safe checks.

Firstly, lets create a CSV Groovy class in the sources root folder of our Fluent SDK project using the following code:

```
// Ensure no package declaration

class CSV {

    List<String> values = (List<String>)[]

    public CSV(String csvString){
        set(csvString)
    }

    public CSV set(String csvString){
        values = (csvString ? csvString.tokenize(',') : (List<String>)[])
        return this
    }

    public String randomValue(){
        if (values.size() > 0) {
            int randomIndex = Math.floor(Math.random() * values.size()).intValue()
            return values.get(randomIndex)
        }
        return ''
    }

    public String toString(){
        return values.join(',')
    }
}
```

Your **src** folder should now look something like this:



Then we can add this file to the **fileIncludes** in our **service-def.json** file:

```
{
  "name": "Welcome Greeting",
  "description": "Return a random greeting",
  "type": "Dynamic Data",
  "version": 1,
```

```
  "tmMinVersion": "5.0.0",
  "parameters": [
    {
      "name": "groovyScript",
      "filePath": "WelcomeGreeting.groovy",
      "fileIncludes": ["../CSV.groovy"],
      "bind": true,
      "required": false,
      "clearOnExport": false,
      "readOnly": true
    },
    {
      "name": "Unit Test Script",
      "filePath": "WelcomeGreetingTest.groovy",
      "fileIncludes": [],
      "bind": false,
      "required": false,
      "clearOnExport": false,
      "readOnly": true,
      "unitTest": true
    },
    {
      "name": "Help Doc",
      "type": "HTML",
      "filePath": "service-help.html",
      "bind": false,
      "required": false,
      "clearOnExport": false,
      "readOnly": true
    },
    {
      "name": "greetingsCsv",
      "description": "A comma separated list of greetings used to generate a greeting at random.",
      "type": "Text",
      "required": true,
      "readOnly": false,
      "value": "Hi,Hello,G'Day,Allo,Hola,Bonjour,Salut,Buenos dias",
      "clearOnExport": false
    }
  ]
}
```

Now we are ready to adjust our service code to utilize this common class by making the following code changes:

Original Code

```
        // Select a greeting from the list at random
        List<String> greetings = svcDef.paramsMap.greetingsCsv.tokenize(',')
        String randomGreeting = greetings.get(Math.floor(Math.random() * greetings.size()).intValue())
```

New Code to Utilize the CSV Class

```
        // Select a greeting from the list at random
        String randomGreeting = new CSV(svcDef.paramsMap.greetingsCsv).randomValue() ?: 'Hi'
```

The resulting service code should now look like this:

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.vo.*
import javax.servlet.http.*
import groovy.json.JsonBuilder

class WelcomeGreeting {

    String invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {

        // Read the persons name from the request
        String personsName = request.getParameter('name') ?: 'friend'

        // Select a greeting from the list at random
```

```
        String randomGreeting = new CSV(svcDef.paramsMap.greetingsCsv).randomValue() ?: 'Hi'

        // Create the result Map containing the greeting
        Map result = [:]
        result.greeting = (randomGreeting + ' ' + personsName)

        // Convert to JSON and return
        return new JsonBuilder(result).toString()
    }
}
```

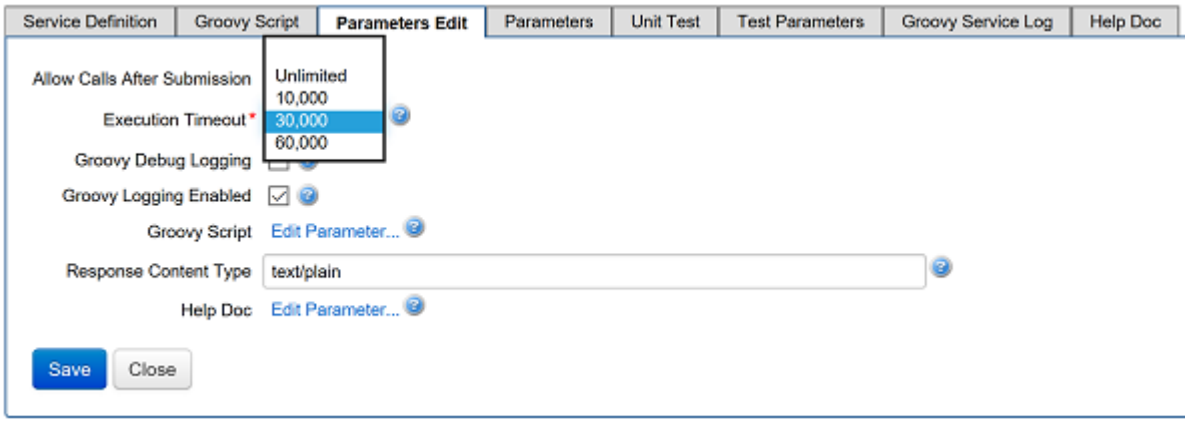
If you deploy this service to Transaction Manager and view the Groovy Script tab via the console you will see that the CSV class has been appended to the bottom of the script:

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.vo.*
import javax.servlet.http.*
import groovy.json.JsonBuilder

class WelcomeGreeting {

    String invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {

        // Read the persons name from the request
        String personsName = request.getParameter('name') ?: 'friend'

        // Select a greeting from the list at random
        String randomGreeting = new CSV(svcDef.paramsMap.greetingsCsv).randomValue() ?: 'Hi'

        // Create the result Map containing the greeting
        Map result = [:]
        result.greeting = (randomGreeting + ' ' + personsName)

        // Convert to JSON and return
        return new JsonBuilder(result).toString()
    }
}

// GPackage Include: ../CSV.groovy
class CSV {

    List<String> values = (List<String>)[]

    public CSV(String csvString){
        set(csvString)
    }

    public CSV set(String csvString){
        values = (csvString ? csvString.tokenize(',') : (List<String>)[])
        return this
    }

    public String randomValue(){
        if (values.size() > 0) {
            int randomIndex = Math.floor(Math.random() * values.size()).intValue()
            return values.get(randomIndex)
        }
        return ''
    }

    public String toString(){
        return values.join(',')
    }
}
```

Managing External Service Timeouts

Sometimes Groovy Script services may not return because of a external system responding extremely slowly, or because of some coding error causing an infinate loop.

To guard against these types of issues, Groovy Services are configured with an execution time out. If the Groovy script has not completed it's execution in the given time it will be interrupted and a GroovyScriptException will be thrown. By default most Transact Groovy Services have a default execution timeout of 1 minute. This value is configured as a service parameter called **`executionTimeout`**.



If a Groovy Service does not define an **`executionTimeout`** then the system Deployment Property 'Groovy Script Timeout' will be used instead. The default 'Groovy Script Timeout' value is 10 minutes.

Techniques for Solving Common Requirements

Performing Remote Service Calls

Transact Groovy services are often used call external REST or SOAP Web Services to perform various integrations. Typical examples of calling remote services include:

- dynamic data services
- form prefill
- delivery processes

Avoka Fluent HTTP

To call external services we recommend using the Fluent HTTP library. This library is extremely easy to use, automatically cleans up resources, and provides good protection against most failure modes which can occur calling remote services.

Avoka Fluent HTTP provides a wrapper around the powerful Apache HttpComponents library. While Apache HTTP Components is incredibly flexible, it can be difficult to use and requires developers to write defensive code to handle the various remoting failure modes.

By default the Avoka Fluent HTTP library automatically configures the following settings:

- 10 second timeout for connection establishment
- 60 second timeout for socket read
- 8 MB maximum response read size
- JVM proxy connection settings if defined

REST GET Example

The example below uses the GetRequest object to perform REST HTTP GET request and returns a Path object. The path object is used to access values from the JSON response content, in this example a nested contact email address.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.http.*
import com.avoka.tm.vo.*
import javax.servlet.http.*

class RestGetService {

    String invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {

         Map parameters = [:]
         parameters.id = request.getParameter('customerId')
         parameters.account = request.getParameter('accountId')

         String username = svcDef.paramsMap.username
         String password = svcDef.paramsMap.password

         // execute GET request and return a HttpResponse object
         HttpResponse response = new GetRequest('https://service.mycorp.com/secure/rest/accounts/')
             .setParams(parameters)
             .setBasicAuth(username, password)
             .execute()

         // ensure response is OK
         if (!response.isStatusOK()) {
             throw new RuntimeException(response.statusLine)
         }

         // get Path object from the response
         Path path = response.getPathContent()

         return path.val("contact.email")
    }
}
```

REST POST Example

The example below uses the PostRequest object to perform REST HTTP POST request and returns a JSON text object.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.http.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import javax.servlet.http.*

class RestPostService {

    String invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {
```

```
        String message = '''{
           "cutomerId": 82881,
           "name": "John Doe",
           "email": "john.doe@gmail.com",
           "age": 42,
           "married": true
        }'''

        String username = svcDef.paramsMap.username
        String password = svcDef.paramsMap.password

        // execute POST request and return a HttpResponse object
        HttpResponse response = new PostRequest('https://service.mycorp.com/secure/rest/accounts/')
              .setMessage(message)
              .setBasicAuth(username, password)
              .execute()

        // ensure response is OK
        if (!response.isStatusOK()) {
            throw new RuntimeException(response.statusLine)
        }

        // get JSON object from the response
        String json = response.getTextContent()

        return json
    }
}
```

SOAP POST Example

The example below uses the PostRequest object to perform SOAP Web Service HTTP POST request and returns a XML Document object. Note the request content type being set to 'text/xml'.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.http.*
import com.avoka.tm.vo.*
import javax.servlet.http.*
import org.w3c.dom.*

class SOAPService {

    String invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {

        String message = '''
        <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <SOAP-ENV:Body>
                <m:transaction-continue xmlns:m="java:com.verid.carbon.integration.datatypes">
                    <settings>
                        <account-name>$accountName</account-name>
                        <mode>$mode</mode>
                        <ruleset>$ruleset</ruleset>
                        <transaction-id>$transID</transaction-id>
                    </settings>
                </m:transaction-continue>
            </SOAP-ENV:Body>
        </SOAP-ENV:Envelope>
        '''

        String username = svcDef.paramsMap.username
        String password = svcDef.paramsMap.password

        // execute POST request and return a HttpResponse object
        HttpResponse response = new PostRequest('https://service.mycorp.com/secure/rest/accounts/')
              .setContentType('text/xml')
              .setMessage(message)
              .setBasicAuth(username, password)
              .execute()

        // ensure response is OK
        if (!response.isStatusOK()) {
            throw new RuntimeException(response.statusLine)
        }
```

```
        // get Document XML object from the response
        Document xml = response.getDocumentContent()

        return xml
    }
}
```

Interrogating Structured Data Paths

This article relates to usage of the com.avoka.tm.util.Path class (available in the Transact Fluent API)

One of the most common tasks when writing Groovy services is to read structured data from either JSON or XML payloads. This is essential if you are calling 3rd party APIs as most APIs return structured data in one of these 2 formats.

The Transact Fluent API contains a very handy utility class that makes interrogating data of this type very easy, also overcoming a limitation of Static Compilation & Type Checking.

JSON

This example demonstrates how the Path utility can be used to interrogate JSON data.

```
String jsonData = '''{
        "locality": { "state": "NSW", "region": "Sydney" },
        "resultCount": 3,
        "resultList": [
            { "brewer": "4 Pines",         "productName": "Kolsch",   "rating": 3.9 },
            { "brewer": "Pirate Life",     "productName": "IIPA",     "rating": 4.6 },
            { "brewer": "Modus Operandi",  "productName": "Red IPA",  "rating": 4.3 }
        ]
    }'''

Path jsonPath = new Path(jsonData)

// Get the value of a json data element
println 'resultCount: ' + jsonPath.val('resultCount')
println 'locality region: ' + jsonPath.val('locality.region')

// Get the list of paths at a specified location
println 'resultList size: ' + jsonPath.paths('resultList.*').size()

// Iterate through a JSON array
println '>> ALL RESULTS'
jsonPath.paths('resultList.*').each{
    println '    ' + it.val('brewer') + ' - ' + it.val('productName') + ' - ' + it.val('rating')
}

// Iterate through a filtered JSON array
int minRating = 4
println '>> Rating > ' + minRating
jsonPath.paths('resultList.*').findAll{ it.val('rating') > minRating }.each{
    println '    ' + it.val('brewer') + ' - ' + it.val('productName') + ' - ' + it.val('rating')
}

// Iterate through an ordered JSON array
println '>> Ordered by rating'
jsonPath.paths('resultList.*').sort{ - it.val('rating') }.eachWithIndex{ it, i ->
    println '    ' + (i+1) + '. ' + it.val('brewer') + ' - ' + it.val('productName') + ' - ' + it.val('rating')
}
```

XML

This example demonstrates using Path the interrogate XML data.

See also: Interrogating XML Content Containing Namespaces

```
String xmlData = '''
    <XmlData>
        <locality><state>NSW</state><region>Sydney</region></locality>
        <resultCount>3</resultCount>
        <resultList>
            <result><brewer>4 Pines</brewer><productName>Kolsch</productName><rating>3.9</rating></result>
            <result><brewer>Pirate Life</brewer><productName>IIPA</productName><rating>4.6</rating></result>
            <result><brewer>Modus Operandi</brewer><productName>Red IPA</productName><rating>4.3</rating><
/result>
        </resultList>
    </XmlData>'''
```

```
Path xmlPath = new Path(xmlData)

// Get the value of a xml data element using XPath
println 'resultCount: ' + xmlPath.val('//resultCount')
println 'locality region: ' + xmlPath.val('//locality/region')

// Get the list of paths at a specified XPath location
println 'resultList size: ' + xmlPath.paths('//resultList/result').size()

// Iterate through an XML array
println '>> ALL RESULTS'
xmlPath.paths('//resultList/result').each{
    println '    ' + it.val('//brewer') + ' - ' + it.val('//productName') + ' - ' + it.val('//rating')
}

// Iterate through a filtered XML array
int minRating = 4
println '>> Rating > ' + minRating
xmlPath.paths('//resultList/result').findAll{ new Double(it.val('//rating')) > minRating }.each{
    println '    ' + it.val('//brewer') + ' - ' + it.val('//productName') + ' - ' + it.val('//rating')
}

// Iterate through an ordered XML array
println '>> Ordered by rating'
xmlPath.paths('//resultList/result').sort{ - new Double(it.val('//rating')) }.eachWithIndex{ it, i ->
    println '    ' + (i+1) + '. ' + it.val('//brewer') + ' - ' + it.val('//productName') + ' - ' + it.val('
//rating')
}
```

POJO

This example demonstrates using the Path utility to interrogate an object graph using Plain Old Java Objects (POJOs). Assume the following classes:

```
class Locality{
    String state, region
}
class Product {
    String brewer, productName
    Double rating
}
class Result {
    Locality locality
    int resultCount
    List<Product> resultList
}
```

The following code snippet shows usage of Path interrogate an object graph consisting of instances of these classes:

```
Result pojoData = new Result()
pojoData.resultCount = 3
pojoData.locality = new Locality(state: "NSW", region: "Sydney")
pojoData.resultList = [
        new Product(brewer: "4 Pines", productName: "Kolsch", rating: 3.9),
        new Product(brewer: "Pirate Life", productName: "IIPA", rating: 4.6),
        new Product(brewer: "Modus Operandi", productName: "Red IPA", rating: 4.3)
]

Path pojoPath = new Path(pojoData)

// Get the value of a pojo variable
println 'resultCount: ' + pojoPath.val('resultCount')
println 'locality region: ' + pojoPath.val('locality.region')

// Get the list of paths at a specified location
println 'resultList size: ' + pojoPath.paths('resultList').size()

// Iterate through an Object array
println '>> ALL RESULTS'
pojoPath.paths('resultList').each {
    println '    ' + it.val('brewer') + ' - ' + it.val('productName') + ' - ' + it.val('rating')
}
```

```
// Iterate through a filtered Object array
int minRating = 4
println '>> Rating > ' + minRating
pojoPath.paths('resultList').findAll { it.val('rating') > minRating }.each {
    println '    ' + it.val('brewer') + ' - ' + it.val('productName') + ' - ' + it.val('rating')
}

// Iterate through an ordered Object array
println '>> Ordered by rating'
pojoPath.paths('resultList').sort { -it.val('rating') }.eachWithIndex { it, i ->
    println '    ' + (i + 1) + '. ' + it.val('brewer') + ' - ' + it.val('productName') + ' - ' + it.val('rating')
}
```

Output

The output from all the above code snippets is exactly the same:

```
resultCount: 3
locality region: Sydney
resultList size: 3
>> ALL RESULTS
    4 Pines - Kolsch - 3.9
    Pirate Life - IIPA - 4.6
    Modus Operandi - Red IPA - 4.3
>> Rating > 4
    Pirate Life - IIPA - 4.6
    Modus Operandi - Red IPA - 4.3
>> Ordered by rating
    1. Pirate Life - IIPA - 4.6
    2. Modus Operandi - Red IPA - 4.3
    3. 4 Pines - Kolsch - 3.9
```

Translating Date Formats

The native Avoka date format is yyyy-MM-dd. This format is used by the forms so all dates coming from forms in either dynamic data calls or submission XML payloads will be in this format.

If 3rd party services require a different date format you will need to reformat the date as required. It may help to write your own helper function to do this E.g.

```
/**
 * Formats an Avoka standard date format string (yyyy-MM-dd) to a greenId format date (dd/MM/yyyy)
 *
 * @param avokaDateString The Avoka standard date format (yyyy-MM-dd) as a string
 * @return the greenId formatted date as string
 */
static String formatDate(avokaDateString){
    if(!avokaDateString){
        return ''
    }
    def avokaDate = Date.parse('yyyy-MM-dd', avokaDateString)
    def formattedDateString = avokaDate.format('dd/MM/yyyy')
    logger.debug 'Date format conversion from ' + avokaDateString + ' to ' + formattedDateString
    return formattedDateString
}
```

Interrogating XML Content Containing Namespaces

So you've written a Groovy service that calls a web service and gets a response back that looks something like this:

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header>
  </env:Header>
  <env:Body>
    <env:Fault xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
      <faultcode>env:Server</faultcode>
      <faultstring>Errors with input data</faultstring>
      <detail>
        <ns2:faultDetails xmlns:ns2="http://schemas.myorg.com/idv/">
          <code>FieldValidationFault</code>
          <details>Invalid Medicare number</details>
        </ns2:faultDetails>
        <ns2:faultDetails xmlns:ns2="http://schemas.myorg.com/idv/">
          <code>FieldValidationFault</code>
          <details>Invalid Passport number</details>
        </ns2:faultDetails>
      </detail>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

You can interrogate this XML structure using the Path utility class with one of the following approaches:

Namespace Notation XPath

The Path class will resolve namespaces automatically for you so you can perform XPath queries using namespace notation:

```
Path xmlPath = new Path(xmlData)
List<Path> subPaths = xmlPath.paths('//ns2:faultDetails')
```

The XPath name() or local-name() Approach

If you're an XPath guru, you will already know that you can structure an XPath query to skirt around the namespace resolution issue using the **local-name** directive. The following code works against the XML above to retrieve the 2 sub paths:

```
List<Path> subPaths = xmlPath.paths('//*[local-name()="faultDetails"]')
```

Alternatively you could use the **name** directive with the namespace included as follows:

```
List<Path> subPaths = xmlPath.paths('//*[name()="ns2:faultDetails"]')
```

This type of query requires a deeper knowledge of the XPath syntax for this more advanced type of query and is a lot less readable than simple queries, but does offer a simple way to achieve the desired result.

Checking for JavaScript Cross-Site Scripting Attacks

If your services are handling GET or POST request parameters and using these in service calls or injection them into XML form prefill data you need to prevent criminals from attacking your solution with JavaScript based attacks.

Transaction Manager provides a Security class to identify XSS values based on the OWASP XSS Filter Evasion Cheat Sheet rules. The script example below is redirecting the user to an Not Authorized page if a URL request parameter contains a dangerous XSS value.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.util.Security
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentFormSecurityFilter {

    String invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {

        def accountId = request.getParameter('accountId');

        if (!Security.isXssSafeText(accountId)) {
            throw new RedirectException('../not-authorized.htm')
        }

        return "";
    }
}
```

Avoiding SQL Injection Attacks

If you are using form data in database code please take care prevent SQL injection attacks. Use prepared statements which will automatically escape text values for you.

```
// Get the contact details out of the form submission XML
def xmlDoc = new XmlDoc(submissionXml)
def firstName = xmlDoc.getText("//Contact/FirstName")
def lastName = xmlDoc.getText("//Contact/LastName")
def email = xmlDoc.getText("//Contact/Email")

// Create a database SQL object
def sql = Sql.newInstance("jdbc:mysql://localhost:3306/mydb", "root", "password", "com.mysql.jdbc.Driver")

// Use a prepare statement to insert values
sql.execute("insert into contact_record (first_name, last_name, email) values (?, ?, ?)", [firstName, lastName,
email])
```

Continuous Integration

Introduction

Continuous Integration is an essential ingredient in a software development process, which helps improve software quality and reduce risk.

The benefits are many, and primarily flow from the implementation of reliable, repeatable, well designed automated processes which save time, reduce the chance of human error, and identify errors early.

This article describes the services and tooling provided by the Avoka Transact platform to facilitate the implementation of Continuous Integration in support of Transact Application development.

What is Continuous Integration?

Continuous Integration is the frequent building and testing of new code changes, assisted by software tooling to automate the process.

Components in a typical CI environment generally include:

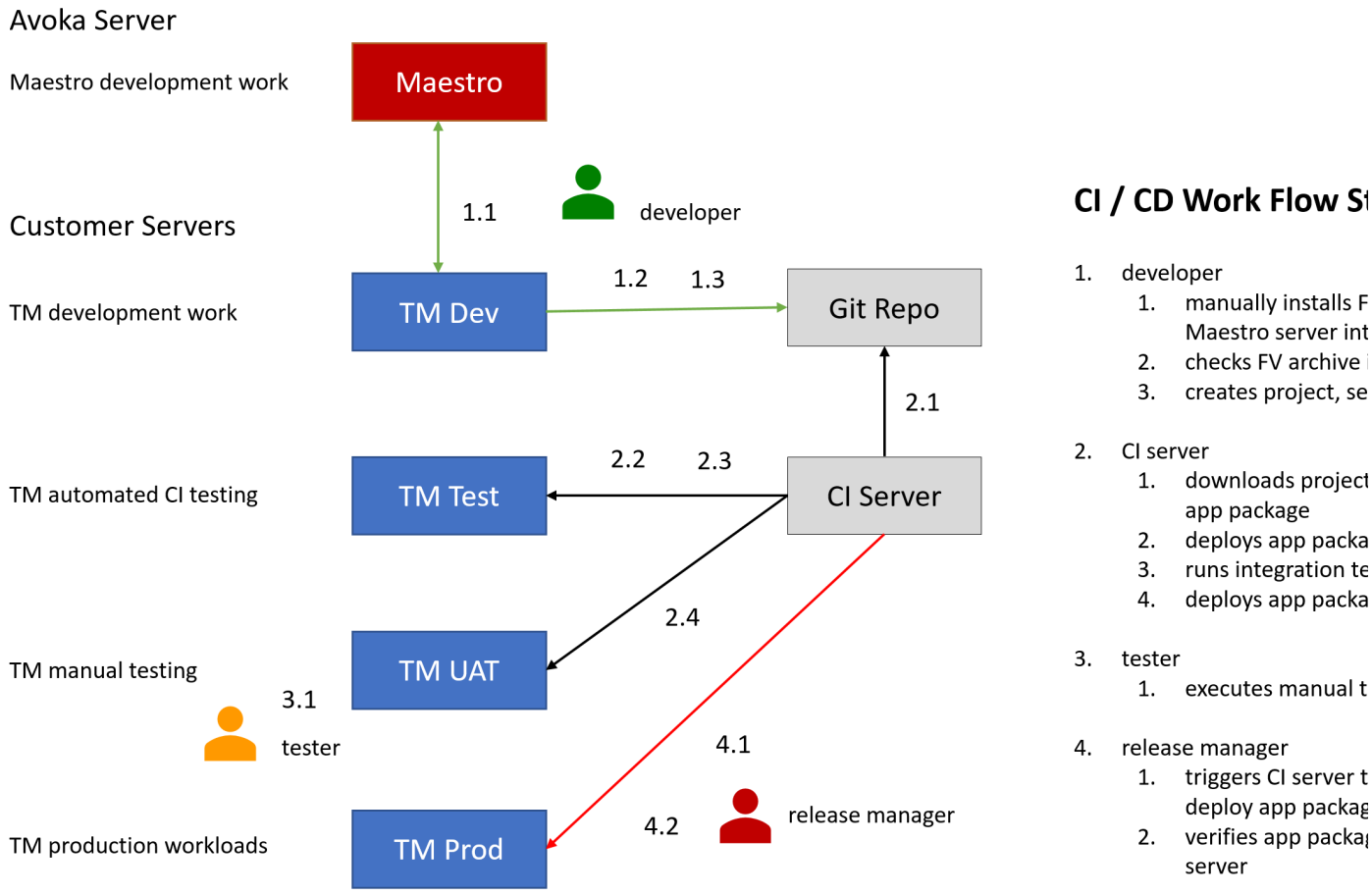
- A Source control system such as Git and Subversion
- Build automation tools such as Ant
- Dependency management tools such as Maven
- CI Servers such as Jenkins
- Unit testing frameworks such as JUnit
- Mocking libraries such as Mockito
- Configuration automation tools such as Ansible and Chef

CI for Transact Manager

Transact Manager supports CI practices through three key features:

- Application Packages - bundles containing primary deployable artifacts such as forms and services
- Transact REST API Reference - which includes services to facilitate the deployment of application packages
- Transact Fluent SDK - which includes tooling in the form of Ant tasks which leverage the REST APIs

The diagram below shows the manual and automated workflow involved in the CI environment.



Application Packages

Application Packages

Application Packages support deployment into Transact Manager instances. They are Zip format files which contain a folder structure and files that define:

- Delivery Channels
- Forms
- Properties
- Service Connections
- Services

An application package is created by the 'appackage' Ant task, which requires 2 parameters:

- Name of definition file - which itself contains references to other files and folders which will be packaged
- Name of output file - which will be the Application Package zip file

Application Package Definition File

The file below shows the content of a sample application package definition file. The main file 'app-package-def.json' contains references to other definition files. The first file from each group is also shown, and where appropriate, screenshot show where the deployed entities are visible in Transact Manager.

Example app-package-def.json file

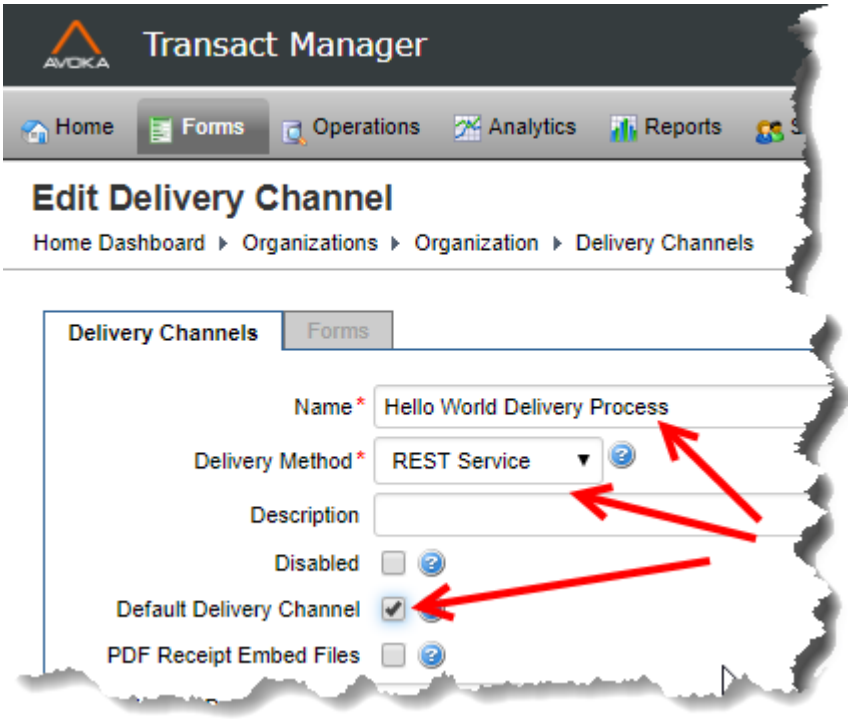
```
{
  "name": "Hello World App package",
  "description": "TODO...",
  "clientCode": "maguire",
  "delivery": [
    "delivery/rest-service-delivery-def.json",
    "delivery/trash-can-delivery-def.json"
  ],
  "forms": [
    "forms/CCA-MAESTRO-AP/cca-maestro-form-def.json",
    "forms/customer-enquiry-ap/customer-enquiry-form-def.json"
  ],
  "properties": [
    "properties/property-def.json",
    "properties/property-de-def.json",
    "properties/property-def1.json"
  ],
  "connections": [
    "connections/connection-def.json",
    "connections/connection-def2.json"
  ],
  "services": [
    "services/helloworld/service-def.json"
  ]
}
```

Delivery Channels

delivery/rest-service-delivery-def.json

```
{
    "name": "Hello World Delivery Process",
    "deliveryMethod": "REST Service",
    "defaultChannel": true,
    "retryDelayMins": 15,
    "serviceName":"Hello World Delivery Process"
}
```

The above delivery service definition will appear in Transact Manager console as:



Forms

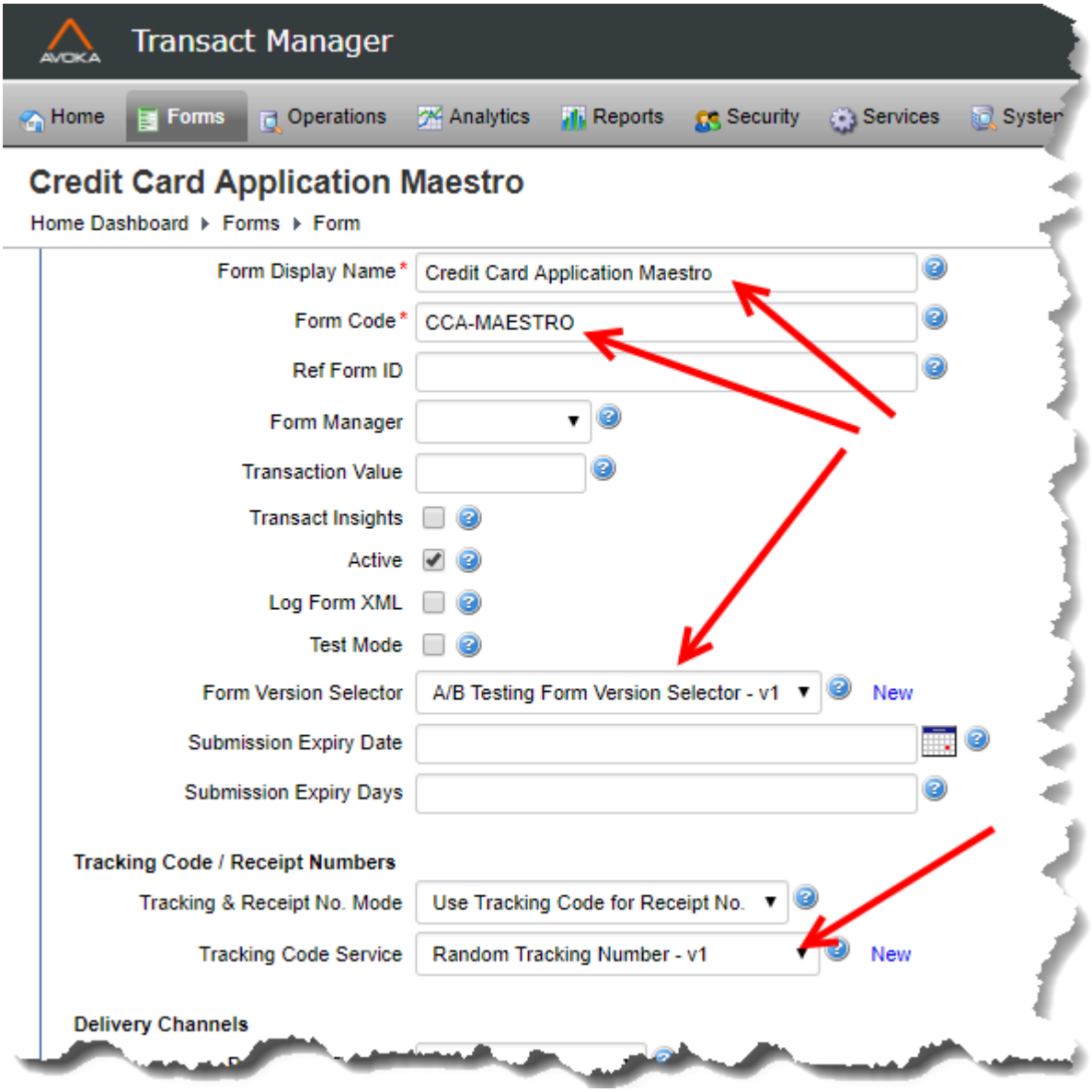
The file below will cause a form to be uploaded into Maestro, and subsequently be visible in the Transact Manager console:

forms/CCA-MAESTRO-AP/cca-maestro-form-def.json

```
{
   "name":"Credit Card Application Maestro",
   "formCode":"CCA-MAESTRO-AP",
   "currentFormVersionNumber":"9",
   "formVersionSelector":{
         "serviceName":"A/B Testing Form Version Selector",
         "serviceVersion":1
   },
   "trackingCodeService":{
         "serviceName":"Random Tracking Number",
         "serviceVersion":1
   },
   "defaultChannels":{
      "productionDelivery":"Hello World Delivery Process",
      "abandonDelivery":"Hello World Delivery Process",
      "validationFailureDelivery":""
   },
   "dataRetention":{
      "savedDays":30,
      "finishedDays":5
   },
   "emailConfirmation":"Confirmation and PDF Receipt",
   "formSpaces":[
      {
         "name":"Maguire",
         "anonAccess":true,
         "authAccess":true
      }
   ],
   "formVersions":[
      {
         "versionNumber":"9",
         "formDataEncryption":true,
         "formType":"Maestro Form",
         "formVersionFile":"./form-version-Credit_Card_Application_Maestro-all-2017-06-26.zip",
         "properties":[
            {
               "name":"Form Description",
               "description":"Form Description",
               "dataType":"String",
               "value":""
            },
```

```
            {
               "name":"Locale De",
               "description":"Form Description",
               "dataType":"JSON",
               "valueFile":"locale-de.json"
            }
         ],

         "formDataConfig":{
            "contactEmailXPath":"//ContactDetails/EmailAddress"
         }
      },
      {
         "versionNumber":"8",
         "formDataEncryption":true,
         "formType":"Maestro Form",
         "formVersionFile":"./form-version-Credit_Card_Application_Maestro-all-2017-06-26.zip",
         "properties":[
            {
               "name":"Form Description",
               "description":"Form Description",
               "dataType":"String",
               "scope":"Form",
               "value":""
            }
         ],

         "formDataConfig":{
            "contactEmailXPath":"//ContactDetails/EmailAddress"
         }
      }
   ]
}
```

The form appears in Transact Manager console as:



When specifying forms, form version archive files need to be manually exported from Transact Manager, saved locally and checked into source control. They can then be referenced by a 'formVersionFile' property in the 'formVersions' array of the form definition Json file as shown above.

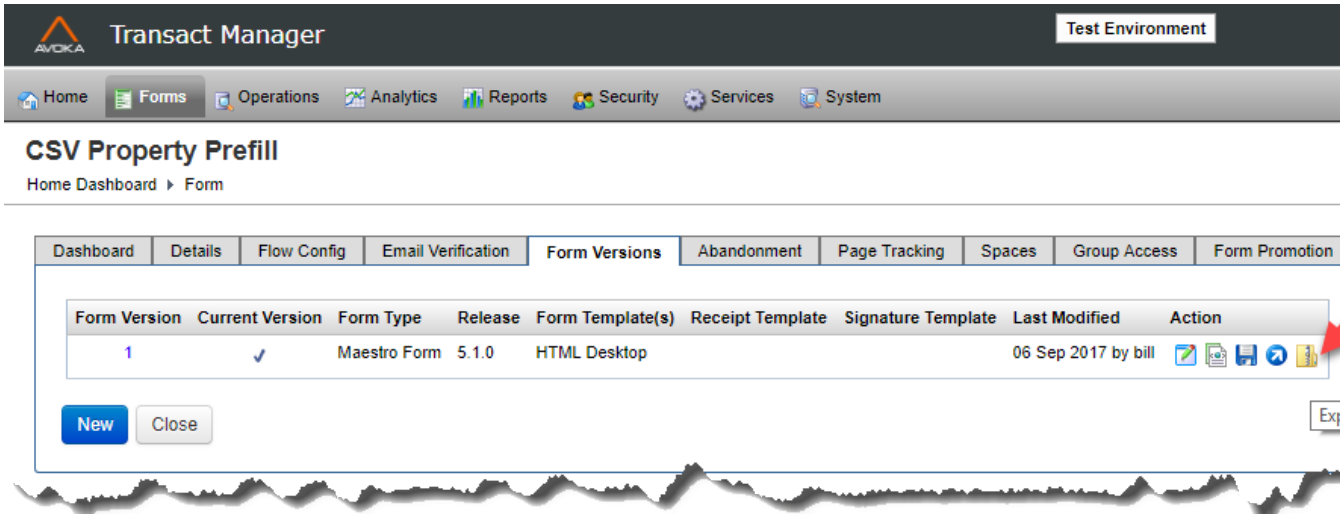
Exporting can be performed in the Transact Manager Form panel, Form Versions tab for a selected form as shown below:

The 'emailConfirmation' property, if specified, must be one of the following values:

- None
- Confirmation
- Confirmation and PDF Receipt
- Link to Receipt

forms/CCA-MAESTRO-AP/locale-de.json

```
{
        "key.de": "Brüderlich zusammenhält"
}
```

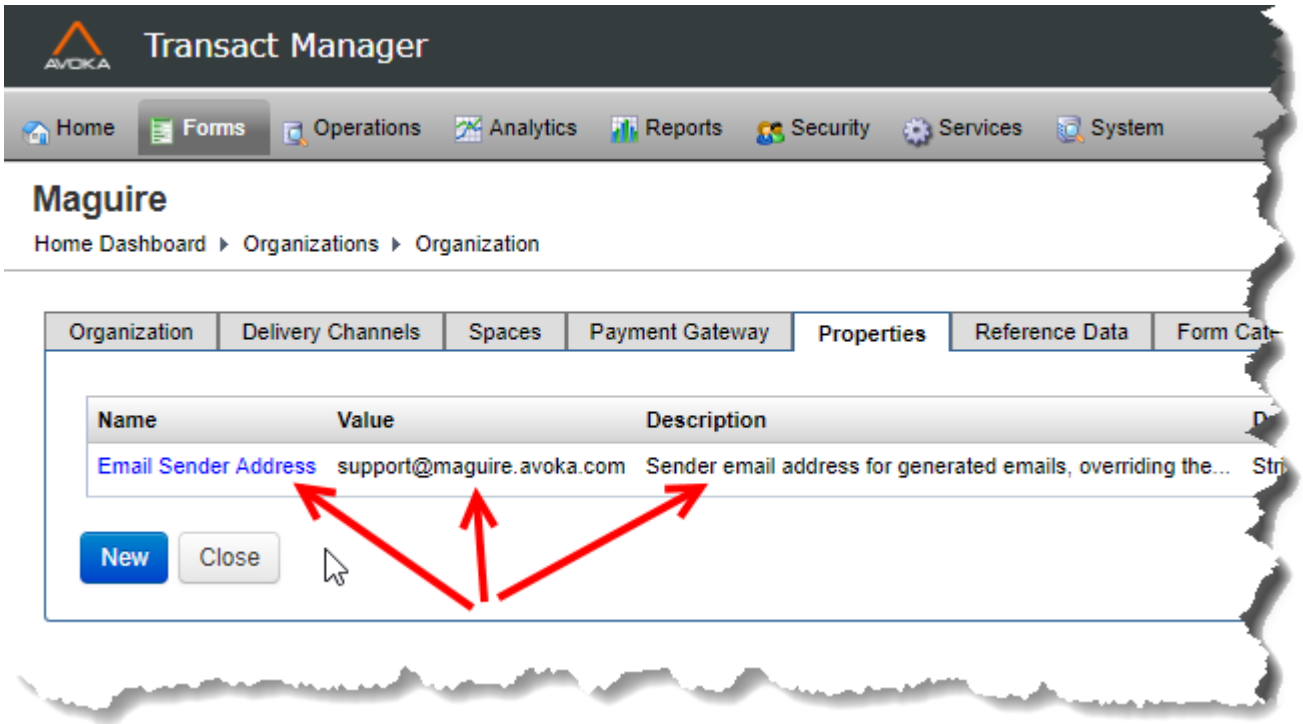


Properties

properties/property-def.json

```
{
        "name": "Email Sender Address",
        "description": "Sender email address for generated emails, overriding the global email sender address.",
        "scope": "Form",
        "dataType": "String",
        "value": "support@maguire.avoka.com"
}
```

The property above will be visible in Transact Manager console as:



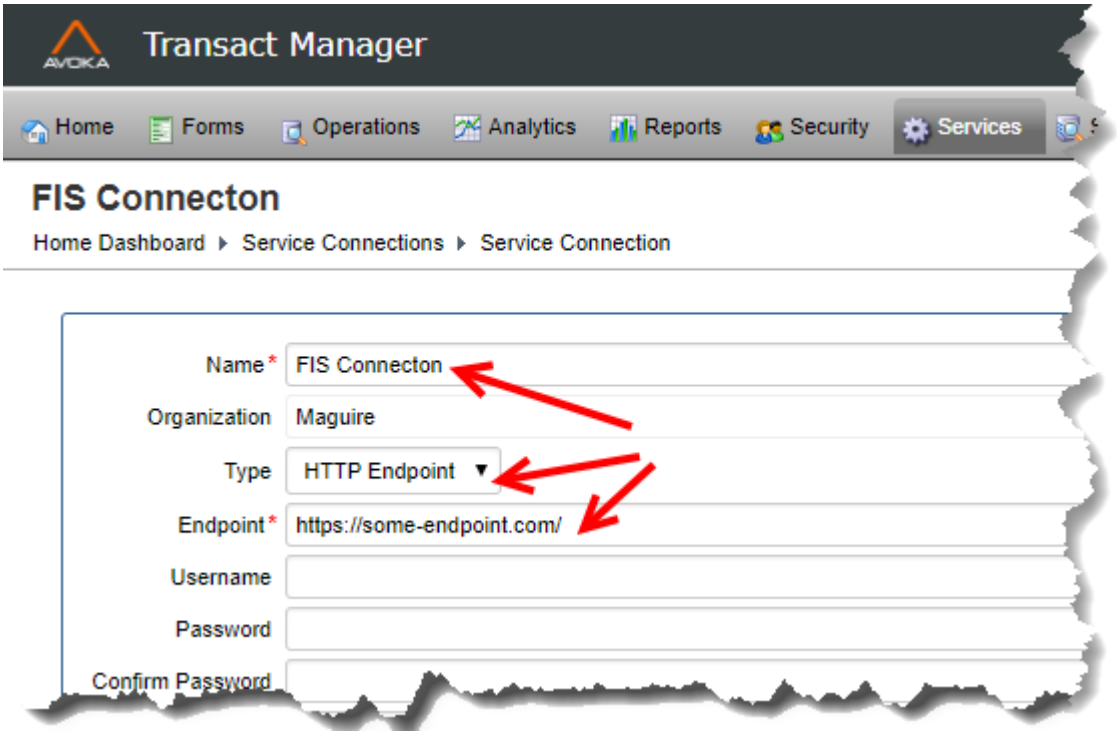
Service Connections

connections/connection-def.json

```
{
   "name": "FIS Connection",
   "type": "HTTP Endpoint",
   "endpoint": "https://some-endpoint.com/",
   "username": "username",
   "password": "password",
```

```
   "dataFile": ""
}
```

After installation of the Application Package, the above Service Connection details will be available to see in the Transact Manager console as shown below:



Services

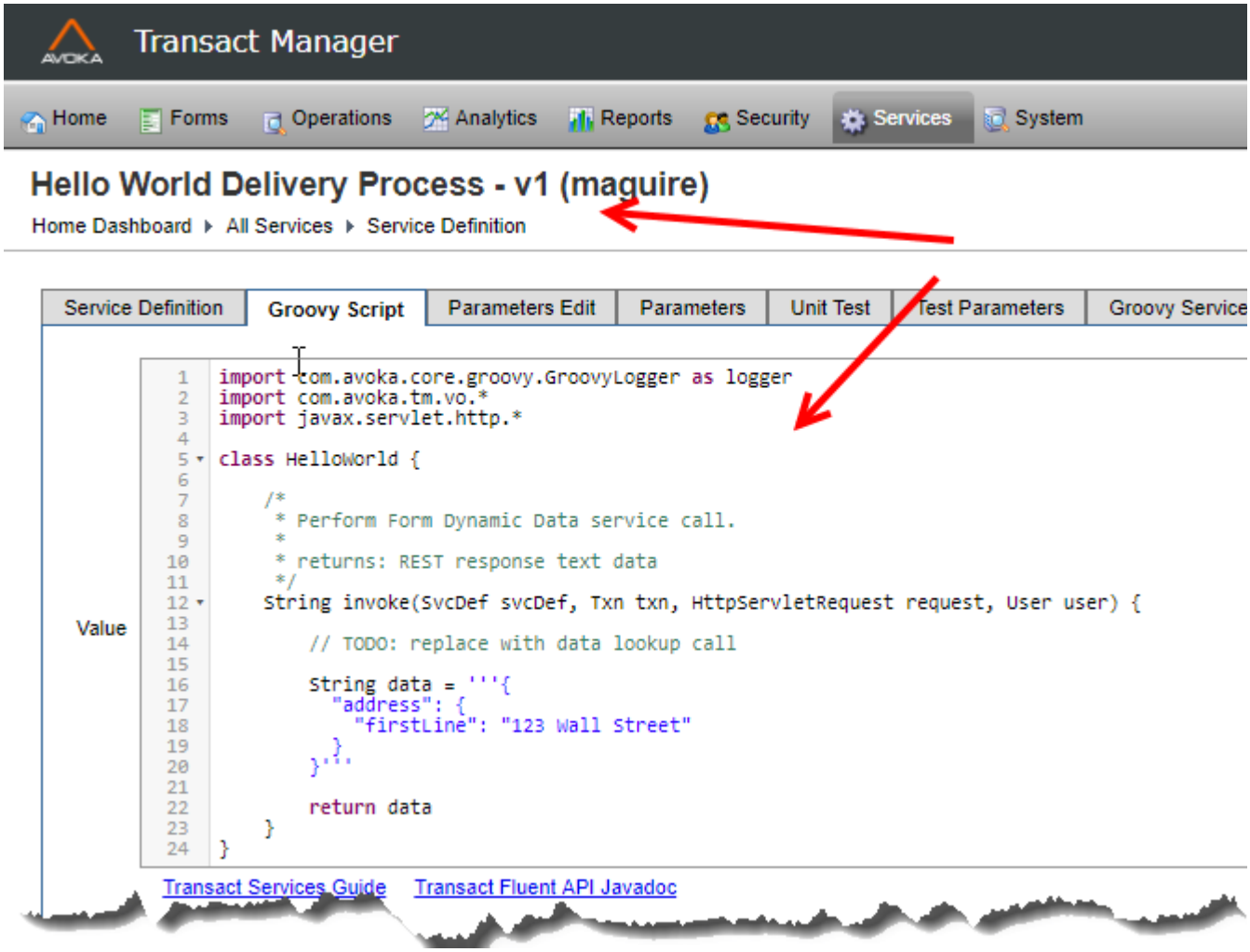
services/helloworld/service-def.json

```
{
  "name": "Hello World Delivery Process",
  "description": "",
  "type": "Delivery Process",
  "version": 1,
  "tmMinVersion": "5.0.0",
  "parameters": [
    {
      "name": "groovyScript",
      "filePath": "HelloWorld.groovy",
      "fileIncludes": [],
      "bind": true,
      "required": false,
      "clearOnExport": false,
      "readOnly": false
    },
    {
      "name": "Unit Test Script",
      "filePath": "HelloWorldTest.groovy",
      "fileIncludes": [],
      "bind": false,
      "required": false,
      "clearOnExport": false,
      "readOnly": false,
      "unitTest": true
    },
    {
      "name": "Help Doc",
      "type": "HTML",
      "filePath": "service-help.html",
      "bind": false,
      "required": false,
      "clearOnExport": false,
      "readOnly": false
    }
  ]
}
```

services/helloworld/HelloWorld.groovy

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.vo.*
import javax.servlet.http.*

class HelloWorld {

    /*
     * Perform Form Dynamic Data service call.
     *
     * returns: REST response text data
     */
    String invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {

        // TODO: replace with data lookup call

        String data = '''{
          "address": {
            "firstLine": "123 Wall Street"
          }
        }'''

        return data
    }
}
```

The two files above will provision Transact Manager with a new service, and the groovy script will be visible in the console as:



Application Package Definition File Reference

This page documents the possible Json properties in Application Package Definition FIles, along with their allowed values.

Property Name	Required	Allowed Values	Notes
Application package definition file			
name		Any string	
description		Any string	
clientCode		Any valid client code	
connections		Array of paths to connection definition files	Connection definition file
delivery		Array of paths to delivery definition files	Delivery definition file
forms		Array of paths to form definition files	Form definition file
properties		Array of paths to property definition files	Property definition file
services		Array of paths to service definition files	Service definition file
Connection definition file			
name	R	String	

type	R	**Allowed Service Connection types**
		`AWS KMS`
		`AWS S3`
		`DocuSign`
		`HTTP Endpoint`
		`SalesForce`
		`TIG`
		`JBoss`
		`WebLogic`
		`WebSphere`	

endpoint		String depending on type	
username		String	
password		String	
fileData			not currently used
Delivery definition file			
name	R	String	
description		Any string	

deliveryMethod	R	**Allowed deliveryMethod values**
		`Delivery Process`
		`REST Service`	

defaultChannel		boolean	
retryDelayMins		integer	
serviceName		String	Only for deliveryMethod Delivery Process
serviceVersion		String	Only for deliveryMethod Delivery Process
Form definition file			
name	R	String	
formCode	R	valid form code string	
currentFormVersionNumber	R	String	

emailConfirmation		**Allowed emailConfirmation values**
		`None`
		`Confirmation`	

		Confirmation and PDF Receipt Link to Receipt	
transactInsights		boolean true/false	
logFormXml		boolean true/false	
deliveryChannels		object	see deliveryChannels
formVersionSelector		object	see formVersionSelector
trackingCodeService		object	see trackingCodeService
formSpaces		Array of objects	see formSpaces
formGroups		Array of objects	see formGroups
formVersions		Array of objects	see formVersions
deliveryChannels (Form definition file)			
abandonDelivery		String	
productionDelivery		String	
formVersionSelector (Form definition file)			
serviceName		String	
serviceVersion		Number	
trackingCodeService (Form definition file)			
serviceName		String	
serviceVersion		Number	
formSpaces (Form definition file)			
name	R	Form space name	
anonAccess	R	Boolean true/false	
authAccess	R	Boolean true/false	
formGroups (Form definition file)			
name	R	Group name	
description		Any string	
reassignTask		Boolean true/false	
savedAssignedForm		Boolean true/false	
completedForm		Boolean true/false	
newForm		Boolean true/false	
workGroupShare		Boolean true/false	
formVersions (Form definition file)			
versionNumber	R	String	
formDataEncryption		boolean true/false	
unifiedAppData		boolean true/false	
formType	R	**Allowed Form Type values** Maestro Form Composer Form	
formVersionFile	R	Path to form version archive file	
formDataConfig		Object See below	see formDataConfig
formFunctions		Array of Objects	see formFunctions
services		Array of Objects - see below	see services
properties		Array of Objects - see below	see properties
formDataConfig (Form definition file... formVersions)			
contactEmailXPath		Xpath string	

saveChallengeXPath		Xpath string	
dataExtractMapping		Array of objects - see below	see dataExtractMapping
requestParamPrefillMapping		Array of objects - see below	see requestParamPrefillMapping
formFunctions (Form definition file... formVersions)			
services (Form definition file... formVersions)			
properties (Form definition file... formVersions)			
dataExtractMapping (Form definition file... formVersions... formDataConfig)			
name		String	
xpath		Xpath string	
searchable		Boolean true/false	
requestParamPrefillMapping (Form definition file... formVersions... formDataConfig)			
paramName		String	
xpath		Xpath string	
Property definition file			
name	R	String	
description		Any string	

dataType	R	**Allowed Property data types**
		`Boolean`
		`CSV`
		`HTML`
		`Image`
		`JSON`
		`List`
		`Long Text`
		`Number`
		`String`	

value		Any string	
valueFile		Path to file	
Service definition file			
name	R	String	
description		Any string	
clientCode		String	
legacyGroovy		boolean true/false	

type	R	**When legacyGroovy is false**
		`Delivery Function`
		`Fluent Function`
		`Fluent Delivery Process`
		`Fluent Delivery Process with Checkpoints`
		`Fluent Dynamic Data`
		`Fluent Email Service`
		`Fluent Form Prefill`
		`Fluent Form Saved Processor`
		`Fluent Form Security Filter`	

		Fluent Form Version Selector Fluent Groovy Service Fluent Groovy Service - REST Data Loader Fluent Job Action Fluent Receipt Number Fluent Render Receipt Fluent Scheduled Service Fluent Submission Preprocessor Fluent Submission Completed Processor Fluent Submission Data Validator Fluent Tracking Number Fluent Task Expiry Fluent Transaction History Publisher Fluent Virus Scan Job Controller **When legacyGroovy is true** Groovy Delivery Process Groovy Delivery Process with Checkpoints Groovy Dynamic Data Groovy Email Service Groovy Form Prefill Groovy Form Saved Processor Groovy Form Security Filter Groovy Form Version Selector Groovy Groovy Service Groovy Groovy Service - REST Data Loader Groovy Job Action Groovy Receipt Number Groovy Render Receipt Groovy Scheduled Service Groovy Submission Preprocessor Groovy Submission Completed Processor Groovy Submission Data Validator Groovy Tracking Number Groovy Task Expiry Groovy Transaction History Publisher Groovy Virus Scan	
version		valid version string	
tmMinVersion		valid version string	
serviceConnection		String	
parameters		Array of objects	see Parameters (Service definition file)

Parameters (Service definition file)

name	R	String	
description		Any string	
type		**Allowed parameter type values** Groovy Script HTML JSON String	
bind		Boolean true/false	only for groovy script
clearOnExport		Boolean true/false	
readOnly		Boolean true/false	
required		Boolean true/false	
unitTest		String	Only for unit test groovy code
filePath		String	Content replaces 'value' property content Runs before fileIncludes array is processed
value		String	Content will be replaced by filePath if present
fileIncludes		Array of file path names	Each file will be appended to the 'value' property

Version Strings

Up to 20 characters formatted major[.minor[.patch[-qual]]]

Transact Fluent SDK

Purpose

The Transact Fluent SDK is intended for use in developers' IDEs. It contains libraries and project templates for all of the different services it supports.

Download

You can download current and previous versions of the Transact Fluent SDK from this link: Transact Fluent SDK Download

REST API

The SDK makes use of services provided by the Transact REST API, which is documented as part of the SDK at this link: Transact REST API Reference

Ant Task Reference

The table below shows the Groovy Ant tasks available for use in your projects:

Ant Task Name	Purpose
svcscaffold	Scaffolds a new Groovy service project from a template. The generated project files can then be edited as required and the project deployed via generated tasks.
svctypecheck	perform script type checking for the given service project
svcpackage	create a service archive ZIP for the given service project
svcupload	upload service archive ZIP to a Transact Manager server
svcunload	delete the specified service definition on the remote Transact Manager server
svctestremote	runs the services unit test on the remote Transact Manager server
apdelete	delete an application package on a Transact Manager server
apget	download an application package from a Transact Manager server
appackage	create an application package archive ZIP for the given project
apscaffold	scaffolds a new application package project, used by a developer as a quick-start
apupload	upload an application package archive ZIP to a Transact Manager server
build	Executes svc-type-check, svc-package, svc-upload, svc-test-remote in order
tpdelete	Deletes a tpac on a Transact Manager server
tpget	Downloads a tpac from a Transact Manager server
tpupload	Uploads a tpac archive ZIP to a Transact Manager server

apscaffold

The apscaffold provides a project scaffolding task to create a new application package project.

When you run this task you specify an application package name and a source directory. The task will then create an `app-package-def.json` file for you, and create an application package project scaffolding, with folder structures component definition files.

Task Parameters

Attribute	Description	Required	Example
name	The application package name.	true	TrackMe
clientCode	The application package organization client code	true	maguire
dir	The target project directory to create and deploy files to.	true	src/trackme-app

Ant Example

The example ant task below will create an application package project using the svcscaffold task. This example will prompt the user to specify the require parameters.

Ant Task

```
<target name="app-package-create" description="Create a new application package">
        <input message="Please enter client code:" addproperty="clientCode"/>
        <input message="Please enter application package name:" addproperty="name"/>
        <input message="Please enter src directory:" addproperty="dir"/>

        <svcscaffold clientCode="${clientCode}" name="${name}" "dir="${dir}"/>
</target>
```

appackage

The "appackage" task will create an application package archive ZIP for the given project.

Task Parameters

Attribute	Description	Required	Example
src	The path to the application definition file.	true	src/trackme-/app-package-def.json
file	The output application package archive ZIP file	true	target/trackme-application-package.zip

Ant Examples

The example ant task below will create an application package archive ZIP for an application package definition specified by the `src` attribute. The `file` attribute specifies the target application package archive ZIP file output.

Ant Task

```
<appackage file="target/trackme-application-package.zip"
          src="src/trackme-/app-package-def.json"/>
```

apget

The apget task will download an application package from a Transact Manager server.

Task Parameters

Attribute	Description	Required	Example
clientCode	The client code of the organization in TM	true	maguire
applicationPackageName	The name of the application package. The application package must belong to the client specified in the "clientCode" parameter.	true	123
url	The TM server URL	true	https://form.mycorp.com/manager/
username	The TM server login username	true	jsmith@mycorp.com
password	The TM server login password	true	password
proxyFile	HTTP proxy configuration properties file location		./proxy.properties
dir	The dir to write application package file	true (if file not specified)	./../downloads/
file	The file path to write application package file	true (if dir not specified)	./../downloads/application-package.zip

TM Security

To use this task the user specified by `username` will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission 'REST Service Definitions API'.

If the user is not a global administrator, only application packages belonging to the organizations assigned to the user will be accessible.

Ant Example

The example Ant task below will delete an application package with OID 14 belonging to a client with client code "mycorp".

Ant Task

```
<apget clientCode="mycorp"
         applicationPackageName="myapp"
         url="https://form.mycorp.com/manager/"
         username="jsmith@mycorp.com"
         password="password"
         dir="./../downloads/"
         proxyFile="./proxy.properties" />
```

Example Ant log output

```
Example Ant log output:
application-package-get:
 [apget] Request: GET https://form.mycorp.com/manager/secure/rest/application-package/v1/mycorp/myapp HTTP/1.1
 [apget] Application Package Downloaded: mycorp/myapp
```

apupload

The apupload task will upload an application package archive ZIP to a Transact Manager server.

Task Parameters

Attribute	Description	Required	Example
Attribute	Description	Required	Example
clientCode	The client code of the organization in TM	true	maguire
file	The application package archive ZIP file	true	target/application-package-archive-My_Application_1-2017-01-01.zip
method	The upload method (use POST if the application package does not yet exist on the target server; otherwise use PUT to update an existing application package)	true	POST
url	The TM server URL	true	https://form.mycorp.com/manager/
username	The TM server login username	true	jsmith@mycorp.com
password	The TM server login password	true	password
proxyFile	HTTP proxy configuration properties file location		./proxy.properties

TM Security

To use this task the user specified by `username` will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission 'REST Service Definitions API'.

If the user is not a global administrator, only application packages belonging to the organizations assigned to the user will be accessible.

Ant Example

The example Ant task below will upload an application package archive to a remote TM server. It uses the method "PUT" to update an existing application.

Ant Task

```
<apupload file="target/application-package-archive-My_Application_1-2017-01-01.zip"
                method="PUT"
          clientCode="mycorp"
          url="https://form.mycorp.com/manager/"
          username="jsmith@mycorp.com"
          password="password"
          proxyFile="./proxy.properties" />
```

Example Ant log output

```
application-package-upload-existing:
 [apupload] Request: PUT http://localhost:9080/manager/secure/rest/application-package/v1/ntgov HTTP/1.1
 [apupload] Response: HTTP/1.1 200 OK
 [apupload] Response: {
 [apupload]   "archiveName": "application-package-archive-My_Application_1-2017-01-01.zip",
 [apupload]   "importMessage": "Application package named 'My Application 1' for organization 'test' was
imported successfully.",
 [apupload]   "importStatus": "Completed",
 [apupload]   "importTime": "2017-02-24T16:56+1100"
 [apupload] }
```

apdelete

The apdelete task will delete an application package on a Transact Manager server.

Task Parameters

Attribute	Description	Required	Example
clientCode	The client code of the organization in TM	true	maguire
applicationPackageName	The name of the application package. The application package must belong to the client specified in the "clientCode" parameter.	true	123
url	The TM server URL	true	https://form.mycorp.com/manager/
username	The TM server login username	true	jsmith@mycorp.com
password	The TM server login password	true	password
proxyFile	HTTP proxy configuration properties file location		./proxy.properties

TM Security

To use this task the user specified by `username` will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission 'REST Service Definitions API'.

If the user is not a global administrator, only application packages belonging to the organizations assigned to the user will be accessible.

Ant Example

The example Ant task below will delete an application package with OID 14 belonging to a client with client code "mycorp".

Ant Task

```
<apdelete clientCode="mycorp"
         applicationPackageName="myapp"
         url="https://form.mycorp.com/manager/"
         username="jsmith@mycorp.com"
         password="password"
         proxyFile="./proxy.properties" />
```

Example Ant log output

```
application-package-delete:
 [apdelete] Request: DELETE https://form.mycorp.com/manager/secure/rest/application-package/v1/mycorp/myapp HTTP
/1.1
 [apdelete] Application Package Deleted: mycorp/myapp
```

svcscaffold

svcscaffold

The svcscaffold provides a project scaffolding task to create a new Groovy service project.

When you run this task you specify a service name, a service type and a source directory. The task will then create a `service-def.json` file for you, and create a Groovy Service source file and Unit Test source file using the Transact Groovy service templates.

Task Parameters

Attribute	Description	Required	Example
Attribute	Description	Required	Example
template	The service template. Must be a supported service definition template	true	Dynamic Data
name	The service name.	true	TrackMe DD
version	The service version number	true	2
classname	The Groovy class name to use instead of a class name derived from the service name	false	TrackMeDynData
dir	The target project directory to create and deploy files to.	true	src/trackme-dd

Ant Example

The example ant task below will create a new groovy service using the svcscaffold task. This example will prompt the user to specify the require parameters.

Ant Task

```
<target name="create" description="Create a new Groovy Service">
        <input message="Please enter service template:" addproperty="template"/>
        <input message="Please enter service name:" addproperty="name"/>
        <input message="Please enter service version:" addproperty="version"/>
        <input message="Please enter src directory:" addproperty="dir"/>

        <svcscaffold template="${template}" name="${name}" version="${version}" "dir="${dir}"/>
</target>
```

svcpackage

The svcpackage task will create a service archive ZIP for the given service project.

Task Parameters

Attribute	Description	Required	Example
src	The path to the service definition file. Specify either a src attribute or fileset element.	true	src/trackme-/service-def.json
fileset	The fileset configuration to package multiple service definitions in the same service archive.	true	`<fileset dir="src"> <include name="** /service-def.json"/> </fileset>`
file	The output service archive ZIP file	true	target/trackme-service.zip
tmVersion	The target TM version for service compatibility packaging. For examples TM servers older than 4.3.4 do not support the groovyDebugLogging bindable service parameter.		4.3.0
debug	If true then a copy of the service archive XML in the target directory.		true

The task can also be configured to ignore the `clearOnExport` service definition parameter which prevents any service parameter values being included in the service archive ZIP file by setting the JVM arg `-DignoreClearOnExport=true`. This option is useful for unit runnning tests with test parameter values which you may not want included in the final service build.

Ant Examples

The example ant task below will create a service archive ZIP for a single service definition specified by the `src` attribute. The `file` attribute specifies the target service archive ZIP file output.

Ant Task

```
<svcpackage file="target/trackme-service.zip"
          src="src/trackme-dd/service-def.json"/>
```

The example ant task below will create a service archive ZIP containing multiple service definition specified by the `filset` attribute. The `file` attribute specifies the target service archive ZIP file output.

Ant Task

```
<svcpackage file="target/trackme-services.zip">
    <fileset dir="src">
        <include name="**/service-def.json"/>
    </fileset>
</svcpackage>
```

With the Ant `fileset` element the `dir` attribute is the base directory for the fileset and the `include` element provides a filter of files to include. With this task ensure the fileset include filter will only select service def JSON files.

svcupload

The svcupload task will upload service archive ZIP to a Transact Manager server.

Task Parameters

Attribute	Description	Required	Example
Attribute	**Description**	**Required**	**Example**
file	The output service archive ZIP file	true	target/trackme-service.zip
url	The TM server URL	true	https://form.mycorp.com/manager/
username	The TM server login username	true	jsmith@mycorp.com
password	The TM server login password	true	password
proxyFile	HTTP proxy configuration properties file location		./proxy.properties

TM Security

To use this task the user specified by `username` will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission 'REST Service Definitions API'.

If the user is not a global administrator, only services belonging to the organizations assigned to the user will be accessible.

Ant Example

The example ant task below will upload a service archive ZIP to remote TM server.

Ant Task

```
<svcupload file="target/trackme-service.zip"
         clientCode="mycorp"
         url="https://form.mycorp.com/manager/"
         username="jsmith@mycorp.com"
         password="password"
         proxyFile="./proxy.properties" />
```

Example Ant log output

```
upload:
[svcupload] Request: POST https://form.mycorp.com/manager/secure/rest/service-definitions/v1/upload-service-
archive/ HTTP/1.1
[svcupload] Response: {
[svcupload]   "archiveName": "trackme-service.zip",
[svcupload]   "importMessage": "Imported TrackMe DD to Organization 'mycorp' successfully.",
[svcupload]   "importStatus": "Completed",
[svcupload]   "importTime": "2016-06-20T22:43+1000"
[svcupload] }
```

svctestremote

The svctestremote task runs the services unit test on the remote Transact Manager server.

Task Parameters

Attribute	Description	Required	Example
src	The service definition file		src/trackme-dd/service-def.json
name	The service name to remove		Trackme DD
version	The service version number		2
url	The TM server URL	true	https://form.mycorp.com/manager/
username	The TM server login username	true	jsmith@mycorp.com
password	The TM server login password	true	password
timeout	The connection and socket read timeout in seconds, default value is 30 seconds		60
junitXmlReport	JUnit XML Report output file		target/TEST-junit.xml
proxyFile	HTTP proxy configuration properties file location		./proxy.properties

TM Security

To use this task the user specified by `username` will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission 'REST Service Definitions API'.

If the user is not a global administrator, only services belonging to the organizations assigned to the user will be accessible.

Ant Example

The example ant task below will upload a service archive ZIP to remote TM server.

Ant Task

```
<svctestremote src="src/trackme-dd/service-def.json"
         clientCode="mycorp"
         url="https://form.mycorp.com/manager/"
         username="jsmith@mycorp.com"
         password="password"
         timeout="60"
         junitXmlReport="target/Test-junit.xml" />
```

Example Ant log output

```
test-remote:

[svctestremote] Request: POST https://form.mycorp.com/manager/secure/rest/service-definitions/v1/run-unit-test/
HTTP/1.1
[svctestremote] JUnit XML Report: C:\projects\transact-services\target\Test-junit.xml
[svctestremote] Success: {
[svctestremote]   "serviceName": "TrackMe DD",
[svctestremote]   "versionNumber": 2,
[svctestremote]   "clientCode": "mycorp",
[svctestremote]   "testStatus": "Success",
[svctestremote]   "message": "Test succeeded in 32 ms"
[svctestremote] }
```

svctypecheck

The svctypecheck task will perform script type checking for the given service project.

Task Parameters

Attribute	Description	Required	Example
src	The service definition file	true	src/trackme-dd/service-def.json
failonerror	Fail on error build flag for type check (default to true)	false	false

Ant Examples

The example ant task below is performing type checking of the specified service.

Ant Task

```
<svctypecheck src="src/trackme-dd/service-def.json"/>
```

Example Ant log output

```
type-check:
[svctypecheck] type check errors in 'TrackMeDDTest.groovy':
[svctypecheck] 32: [Static type checking] - Cannot find matching method com.avoka.tm.util.Path#value(java.lang.
String). Please check if the declared type is right and if the method exists.
[svctypecheck]  @ line 32, column 37.
[svctypecheck]            assert "123 Wall Street" == path.value("address.firstLine")
[svctypecheck]                                        ^
[svctypecheck] Completed Type Check
```

svcunload

The svcunload task will delete the specified service definition on the remote Transact Manager server.

This task may be useful when you need to remove a service which is not working on a Transact Manager server.

Task Parameters

Attribute	Description	Required	Example
src	The service definition file		src/trackme-dd/service-def.json
name	The service name to remove		Trackme DD
version	The service version number		2
url	The TM server URL	true	https://form.mycorp.com/manager/
username	The TM server login username	true	jsmith@mycorp.com
password	The TM server login password	true	password
proxyFile	HTTP proxy configuration properties file location		./proxy.properties

TM Security

To use this task the user specified by `username` will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission 'REST Service Definitions API'.

If the user is not a global administrator, only services belonging to the organizations assigned to the user will be accessible.

Ant Example

The example ant task below will delete a service definition on the remote TM server.

Ant Task

```
<svcunload src="src/trackme-dd/service-def.json"
         clientCode="mycorp"
         url="https://form.mycorp.com/manager/"
         username="jsmith@mycorp.com"
         password="password" />
```

Example Ant log output

```
test-remote:
[svctestremote] Request: DELETE https://form.mycorp.com/manager/secure/rest/service-definitions/v1/TrackMe DD/2
HTTP/1.1
[svctestremote] Success: {
[svctestremote]   "serviceName": "TrackMe DD",
[svctestremote]   "versionNumber": 2,
[svctestremote]   "testStatus": "Success",
[svctestremote]   "message": "Test succeeded in 32 ms"
[svctestremote] }
```

tpdelete

The svctestremote task runs the services unit test on the remote Transact Manager server.

Task Parameters

Attribute	Description	Required	Example
clientCode	The client code of the organization in TM	true	maguire
projectName	The project name. The project name must belong to the client specified in the "clientCode" parameter.	true	tpac-project-name
version	The version. The version must belong to the project name specified in the "projectName" parameter.	true	1.0
url	The TM server URL	true	https://form.mycorp.com/manager/
username	The TM server login username	true	jsmith@mycorp.com
password	The TM server login password	true	password
proxyFile	HTTP proxy configuration properties file location		./proxy.properties

TM Security

To use this task the user specified by username will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission 'REST Service Definitions API'.

If the user is not a global administrator, only tpac belonging to the organizations assigned to the user will be accessible.

Ant Example

The example Ant task below will delete a tpac with project name "tpac-project-name" and version "1.0" belonging to a client with client code "mycorp".

Ant Task

```
Ant Task
```

Example Ant log output

tpget

The svctestremote task runs the services unit test on the remote Transact Manager server.

Task Parameters

Attribute	Description	Required	Example
clientCode	The client code of the organization in TM		tpac-project-name
projectName	The project name of the tpac. The tpac must belong to the client specified in the "clientCode" parameter.		maguire
version	The version of the project. The version must belong to the client specified in the "projectName" parameter.		1.0
url	The TM server URL	true	https://form.mycorp.com/manager/
username	The TM server login username	true	jsmith@mycorp.com
password	The TM server login password	true	password
proxyFile	HTTP proxy configuration properties file location		./proxy.properties
dir	The dir to write tpac file		./../downloads/
file	The file path to write tac file		./../downloads/tpac.zip

TM Security

To use this task the user specified by `username` will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission 'REST Service Definitions API'.

If the user is not a global administrator, only services belonging to the organizations assigned to the user will be accessible.

Ant Example

The example Ant task below will download a tpac with project name "tpac-project-name" and version "1.0" belonging to a client with client code "mycorp".

Ant Task

```
<tpget clientCode="mycorp"
         projectName="tpac-project-name"
         version="1.0"
         url="https://form.mycorp.com/manager/"
         username="jsmith@mycorp.com"
         password="password"
         dir="./../downloads/"
         proxyFile="./proxy.properties" />
```

Example Ant log output:

Example Ant log output

```
tpac-get:
 [tpget] Request: GET https://form.mycorp.com/manager/secure/rest/tpac/v1/mycorp/tpac-project-name/1.0 HTTP/1.1
 [tpget] TPac "mycorp/tpac-project-name" downloaded to: ./downloads/tpac-archive-project-name-2017-05-24.zip
```

tpupload

The svctestremote task runs the services unit test on the remote Transact Manager server.

Task Parameters

Attribute	Description	Required	Example
clientCode	The client code of the organization in TM	true	maguire
file	The tpac archive ZIP file	true	target/tpac-fis-chexsystems-v1-0-001.zip
override	The override flag	false	true
url	The TM server URL	true	https://form.mycorp.com/manager/
username	The TM server login username	true	jsmith@mycorp.com
password	The TM server login password	true	password
proxyFile	HTTP proxy configuration properties file location		./proxy.properties

TM Security

To use this task the user specified by `username` will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission 'REST Service Definitions API'.

If the user is not a global administrator, only services belonging to the organizations assigned to the user will be accessible.

Ant Example

The example Ant task below will upload a tpac archive to a remote TM server.

Ant Task

```
<tpupload file="target/tpac-fis-chexsystems-v1-0-001.zip"
         override="true"
         clientCode="mycorp"
         url="https://form.mycorp.com/manager/"
         username="jsmith@mycorp.com"
         password="password"
         proxyFile="./proxy.properties" />
```

Example Ant log output:

Example Ant log output

```
tpac-upload:
 [tpupload] Request: POST http://localhost:9080/manager/secure/rest/tpac/v1/mycorp HTTP/1.1
 [tpupload] Response: HTTP/1.1 200 OK
 [tpupload] Response: {
 [tpupload]   "archiveName": "tpac-fis-chexsystems-v1-0-001.zip",
 [tpupload]   "importMessage": "Imported services for application package 'FIS ChexSystems' for organization
'mycorp' to Organization 'maguire' successfully.",
 [tpupload]   "importStatus": "Completed",
 [tpupload]   "importTime": "2017-02-24T16:56+1100"
 [tpupload] }
```

Ant Build Example

Ant Build Example

An example Ant build.xml project file is provided below. Your build file needs to include a `taskdef` element which references the transact Ant Jar file.

This makes the Groovy Ant tasks available for use in for your project.

build.xml

```
<?xml version="1.0" encoding="utf-8"?>
<project name="transact-sdk" basedir="." default="build">

    <taskdef resource="com/avoka/transact/ant/antlib.xml" classpath="lib/transact-ant-5.1.4.jar"/>

    <property file="build.properties"/>

    <property name="template" value="Dynamic Data"/>
    <property name="name" value="Hello World"/>
    <property name="version" value="1"/>
    <property name="src.dir" value="src/services/helloworld"/>
    <property name="archive.file" value="target/helloworld-v${version}.zip"/>

    <target name="scaffold">
        <svcscaffold template="${template}"
            name="${name}"
            version="${version}"
            dir="${src.dir}" />
    </target>

    <target name="type-check">
        <svctypecheck src="${src.dir}/service-def.json" />
    </target>

    <target name="package">
        <svcpackage src="${src.dir}/service-def.json"
            file="${archive.file}"
            debug="false"/>
    </target>

    <target name="upload">
        <svcupload file="${archive.file}"
            clientCode="${manager.clientCode}"
            url="${manager.url}"
            username="${manager.username}"
            password="${manager.password}" />
    </target>

    <target name="unload">
        <svcunload src="${src.dir}/service-def.json"
            clientCode="${manager.clientCode}"
            url="${manager.url}"
            username="${manager.username}"
            password="${manager.password}" />
    </target>

    <target name="test-remote">
        <svctestremote src="${src.dir}/service-def.json"
            clientCode="${manager.clientCode}"
            url="${manager.url}"
            username="${manager.username}"
            password="${manager.password}"
            junitXmlReport="target/TEST-junit.xml" />
    </target>

    <target name="build" depends="type-check, package, upload, test-remote">
    </target>

</project>
```

The above build.xml file references a build.properties file which is provided below. This properties file should contain all the environment specific build attributes. You would generally not check this file into source control as it may contains sensitive user credentials.

build.properties

```
# TM Environment Specific Properties

# Generally you would not check this file into source control

# TM Server URL
manager.url=http://localhost:9080/manager/

# TM REST User Login Name
manager.username=administrator

# TM REST User Password
manager.password=password

# TM Organization Client Code
manager.clientCode=maguire

An example of proxy.properties file is provided below.
# HTTP Proxy settings
# - settings will be used if "http.proxyHost" value is not blank
# - NTLM configuration will be used if "http.proxyDomain" value is not blank
# - username/password values will be used if "http.proxyUser" value is not blank

## HTTP Proxy host. e.g "proxy.mycompany.com"
http.proxyHost=proxy

## HTTP Proxy port. e.g "8080"
http.proxyPort=8080

## HTTP Proxy domain. e.g "MYCOMPANY"
http.proxyDomain=

## HTTP Proxy workstation. e.g "workstation"
http.proxyWorkstation=

## HTTP Proxy user. e.g "jhondoe"
http.proxyUser=

## HTTP Proxy user. e.g "password"
http.proxyPassword=
```

Getting Started with CI

Continuous environments are implemented using software tools that work alone or together to support the creation of pipelines. Implementation is usually achieved via configuration, rather than requiring specific development. Sometimes, as is the case with Avoka's Transact REST API, support services might need to be provided in order to provide application-specific deployment or configuration.

Common tools for implementing pipelines include:

- A version control system - such as Git, Subversion, Visual Studio Team Services and others
- A build system - such as Ant, Gradle, Maven, including dependency management and packaging tools
- A Continuous Integration system - such as Jenkins, TeamCity, Bamboo, GitLab CI and others
- Task automation tools - such as Ant, Maven, Gradle, Gulp, Make and others
- Deployment and Configuration tools - such as 'Desired State Configuration' tools like Ansible, Chef and others
- In-house APIs - such as the REST APIs built into Transact Manager

There is often crossover in functionality between tools, which blurs the functional boundaries. In GitLab for example, version control, build and CI services are all available in a single-source system.

Tools

Avoka does not prescribe the use of specific tools in order to apply CI techniques to the Transact Platform application development process. Customers are free to configure their existing Ant-compatible CI systems with Ant tasks from the Transact Fluent SDK to provide the underlying automation. For those customers who do not have existing CI infrastructure, and wish to be guided in setting up a new environment for themselves, we recommend they explore these tools:

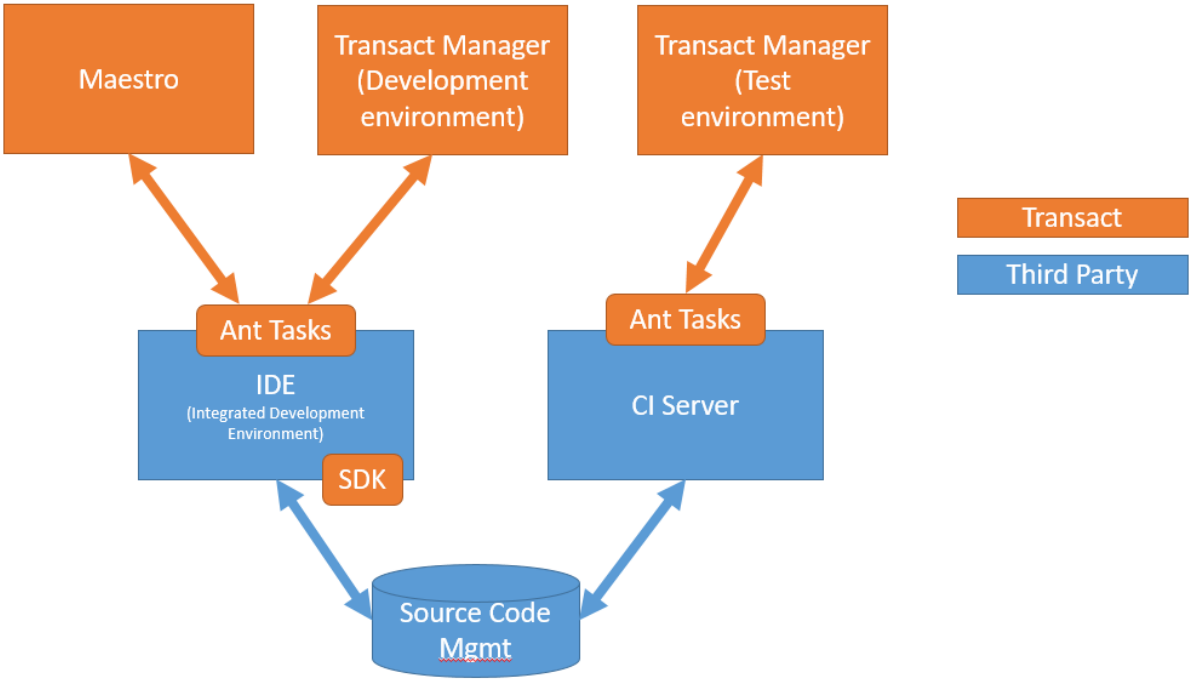
Gitlab - for version control, and issue management if desired - see: https://about.gitlab.com/

Maven - for dependency management of third-party content, and potentially to store output from the build system - see https://maven.apache.org/

Jenkins - for integration to version control systems and orchestration of build, dependency management and CI services - see https://jenkins.io/

Avoka's CI Implementation Model

Avoka's CI implementation is represented in this diagram:



In this model, Transact Application developers using Maestro, Transact Manager console, and other IDEs collaborate in a development environment. They work with assets including forms, templates, organization properties, forms, delivery channels, services, and service connections. They work with additional JSON files that define the contents of Application Packages. Changes in the development environment are committed to version control, from where they can be checked out, packaged into Application Packages and deployed to other environments by Ant tasks interacting with the Transact REST APIs. The scope of this activity includes:

- Checking assets and supporting files into and out of version control
- Triggering CI pipelines that:
    - Execute Ant tasks that form part of the Transact Fluent SDK which:
        - Package assets into 'Application Packages'
        - Publish Application Packages to Transact Manager services exposed via the Transact REST API

Workflow

The workflow involved in implementing a Transact Manager CI workflow involves these steps:

- Acquire and install the Fluent SDK from the download page at Transact Fluent SDK Download
- Define an Application Package
    - Create configuration folder structure
    - Obtain Form Version Archive files from Transact Manager if required
    - Create Json configuration files to describe contents
- Configure your choice of CI server with Ant Tasks from the SDK. To see how this is done using Jenkins, refer to Jenkins Tutorial
- Create your CI workflow, including one or more steps to:
    - Build the application package
    - Deploy the application package to one or more servers
    - Run the remote tests on deployed servers
- Configure the CI server to work with source control. To see how to do this for GitLab, refer to 5-minute GitLab Tutorial
- Check the Application Package configuration files and resources into source control
- Check the state of the CI workflow in the CI server

Version Control System

A version control system is the primary entry point for new code into a system. Continuous integration pipelines begin with detecting changes to source.

Triggers

The trigger for initiating most Continuous Integration pipelines is the detection of source code changes as a result of check-ins to the version control system.

Source code changes could be detected by monitoring the version control system via polling, or triggered from within the version control system via broadcasts or notifications. Whichever method is used is determined by personal preference, and the capability of the systems in use.

Once triggered, the CI pipeline will commence collecting all required assets from source control. Once the collection is complete, the build system can be triggered to gather additional dependencies, create the artifacts required for deployment, and run initial tests.

What goes into version control?

As with any software development project, the version control system should hold assets required to rebuild entire projects from scratch, including:

- Source code - in the case of Transact Manager application development, this could be source for Groovy, Java or JavaScript for example
- Configuration files - typically JSON, XML and/or property files
- Object Libraries - in the absence of a dependency management system such as Maven, third party libraries should be kept in source control
- Resources - including images, icons or other visual components
- Custom content - proprietary format content, including Maestro resources
- Scripts - shell or other scripts that might be used to create folders, create databases and seed tables, configure services or any other actions needed to prepare an execution environment

What doesn't go into version control?

You don't want to store unprotected sensitive credential information in version control where it can be easily viewed. User Ids and especially passwords do not belong in unprotected version control.

Either don't store such content in the first place, or use some kind of encryption or vault technology to secure the content.

It's beyond the scope of this document to go into detail about how to protect sensitive content, but some interesting tools in this area are:

- https://github.com/StackExchange/blackbox
- https://github.com/AGWA/git-crypt
- http://docs.ansible.com/ansible/latest/playbooks_vault.html
- https://github.com/hashicorp/vault
- https://square.github.io/keywhiz/

CI Server

Configuration Tasks

It will typically be necessary to configure the CI server with:

- Java JDK
- Ant installation
- Ant tasks
- build.xml file from the Transact Fluent SDK
- Version control system connection
- Dependency Management System

More advanced configuration might be required to configure triggers between the version control system and the CI server. In the case of GitLab and Jenkins, refer to: https://docs.gitlab.com/ee/integration/jenkins.html

Build System

A build system in a CI pipeline is responsible for converting source assets into deployable artifacts. This could involve compiling source code into executable object code, or some other form of conversion from an input representation to an output object.

A build is typically triggered by one or more of:

- Manual control - for example, running a script at a command prompt, or clicking a button in a web page
- Schedule - a periodic trigger, such as 5pm daily
- Event - something happening such as a check-in to a version control system

A build system can be a standalone system, or perhaps integrated into the version control system, dependency management system, or the CI system.

Dependency Management System

A dependency management system such as Maven, backed by a repository system, is the correct place to store and manage third party artifacts required by a build.

It is also an appropriate system in which to store output from the build.

Application Package CI Example

Scenario

This guide will show step-by-step instructions for preparing to implement an Application Package-based CI workflow, reflecting the following scenario:

1. A new form to collect street addresses is required
2. A new dynamic data service to prepopulate the street address on the form is required
3. The form and service need to be deployed to a test TM server

Manual Setup Tasks

We will accomplish this requirement initially by:

1. Creating a new form in Maestro
2. Downloading the form version archive file from Maestro to the filesystem
3. Using the Fluent SDK to:
    a. scaffold a new project workspace
    b. scaffold a new Application Package in the workspace
    c. scaffold a new service
4. Editing the Application Package definition file to include the form and the service
5. Using the Fluent SDK to:
    a. build the Application Package
    b. deploy the Application Package to a TM server
6. Checking the project workspace into source control

Fluent SDK Step-by-step

We will install and use the Transact Fluent SDK to help create parts of our application. We will use the SDK's scaffolding capability to generate templates for our service and Application Package. Then we will use the SDK's packaging and deployment capabilities to build and install our artifacts on TM servers. Our steps will include:

1. Extracting the Fluent SDK and Project Template distribution files
2. Scaffolding a new project build environment
3. Configuring credentials in transact-auth.properties
4. Configuring and creating a new Application Package
5. Creating a new Maestro form with a call to a dynamic data service
6. Downloading the Maestro form version archive file
7. Creating a new form-def.json file to describe the form
8. Editing the Application Package definition file to include the form-def file
9. Configuring and creating a new dynamic data service
10. Moving the service into our Application Package folder structure
11. Editing the Application Package definition file to include the service-def file
12. Building the Application Package
13. Deploying the Application Package
14. Undeploying the Application Package

Pre-requisites

Before you can use the Fluent SDK, you need to have a working installation of a Java 1.8+ JDK, and Ant installed on your system.

If you want to see how you can install a JDK and Ant without administrator privileges, follow the instructions at Java & Ant

If you have an IDE, you can incorporate the SDK into your IDE, but for this exercise, we will just use the SDK Ant tasks from a terminal shell prompt. If you are using Windows, you can download and install GitBash from https://git-for-windows.github.io/

To follow along with the instructions here, your PATH should be set so that you can execute both java and ant from the command line.

If you are using GitBash and have downloaded and extracted Zip distributions of a JDK into C:\dev\jdk\jdk1.8.0_144, and Ant into C:\dev\Ant\apache-ant-1.10.1, you should setup your environment and PATH as follows:

GitBash Environment Setup

```
export JAVA_HOME=/c/dev/jdk/jdk1.8.0_144
export ANT_HOME=/c/dev/Ant/apache-ant-1.10.1
export PATH=${JAVA_HOME}/bin:${ANT_HOME}/bin:${PATH}
```

Notes on the Ant build files

Ant reads from a 'build' file, and executes a 'task' defined within that file. Ant therefore needs to know which build file to use, and which task to execute.

If you do not tell Ant which build file to use, it uses the default build file 'build.xml'. If there is no build.xml file, you need to tell Ant which file to use with the -f parameter.

If you do not tell Ant which task to execute, it will execute the one defined as default in the build file.

In summary, these are the common variations of the Ant command line:

- ant - executes the default task from the default file 'build.xml'
- ant taskname - executes the 'taskname' task from build.xml
- ant -f buildfilename - executes the default task from the file 'buildfilename'
- ant -f buildfilename taskname - executes the 'taskname' task from the file 'buildfilename'

When you initially unpack the Fluent SDK project template, there is only one Ant build file, 'scaffold.xml'. The default task within 'scaffold.xml' is 'scaffold'. That task scaffolds a new project environment, by creating additional folders and files, including a 'build.xml' file that is used for further asset creation, packaging and deployment.

In general, tasks in scaffold.xml simply copy template files and folders into your workspace. Tasks in build.xml create more complex assets by applying values from properties files.

If you create a new Application Package or service, the create task produces another Ant build file named after the new asset. That build file contains tasks to perform additional processing specific to the asset, such as packaging or deploying it.

Steps

Instructions	Commands and expected output

Open a command prompt for executing commands

We are using GitBash on Windows as it conveniently provides all of the commands requried for this exercise. Use C:\dev as your workspace root folder for th you want your output to exactly match the examples below, although you can use any folder as the root since all of the commands and configuration files sho use relative path references.

Install Fluent SDK

This step unpacks the two distribution files side-by-side into a folder which will become your project workspace.

Obtain the 2 package files from from Transact Fluent SDK Download and save them in your workspace root folder

Extract the API zip into a child folder of the same name as the .zip file, without the .zip extension.

Extract the project template zip into a child folder name of your choice. We use addressApp here.

Placing the two packages side-by-side like this enables the project template to correctly locate dependencies in the API distribution.

Commands

```
unzip -q transact-fluent-api-5.1.7.zip -d transact-fluent-api-5.1.7
unzip -q transact-project-template-5.1.7.zip -d addressApp
ls -l
```

Output

```
$ unzip -q transact-fluent-api-5.1.7.zip -d transact-fluent-api-5.1.7
xxxxxx@XXXXXXXXXX MINGW64 /c/dev/addressApp
$ unzip -q transact-project-template-5.1.7.zip -d addressApp
xxxxxx@XXXXXXXXXX MINGW64 /c/dev/addressApp
$ ls -l
total 25232
drwxr-xr-x 1 xxxxxx 1049089        0 Sep 22 08:23 transact-fluent-api-5.1.7/
-rw-r--r-- 1 xxxxxx 1049089 25824530 Sep 22 08:22 transact-fluent-api-5.1.7.zip
drwxr-xr-x 1 xxxxxx 1049089        0 Sep 22 08:23 addressApp/
-rw-r--r-- 1 xxxxxx 1049089     5726 Sep 22 08:22 transact-project-template-5.1.7.zip
```

Change directories into the project template folder

You are now ready to start using the Fluent SDK

Commands

```
cd addressApp
```

Output

```
$ cd addressApp
xxxxxx@XXXXXXXXXX MINGW64 /c/dev/addressApp
```

Scaffold a new project build environment

When you initially unpack the project template into a folder, it contains only enough files to enable the folder to be further prepared for development tasks. Th scaffolds the files and folders in the current directory that are used for subsequent asset creation and build tasks

Issue the Ant command opposite

It executes the default task from scaffold.xml

The default task will create more files and folders in the workspace which we use in subsequent steps.

Commands

```
ant -f scaffold.xml
```

Output

```
$ ant -f scaffold.xml
Buildfile: C:\dev\addressApp\scaffold.xml

scaffold-build-xml:
     [copy] Copying 1 file to C:\dev\addressApp

scaffold-build-properties:
     [copy] Copying 1 file to C:\dev\addressApp
[propertyfile] Updating property file: C:\dev\addressApp\tmp-build.properties
     [move] Moving 1 file to C:\dev\addressApp

scaffold-proxy-properties:
     [copy] Copying 1 file to C:\dev\addressApp

scaffold-package-dir:
     [copy] Copying 3 files to C:\dev\addressApp\package

scaffold-src-dir:
     [copy] Copying 2 files to C:\dev\addressApp\src

scaffold-new-service-properties:
     [copy] Copying 1 file to C:\dev\addressApp

scaffold-maestro-properties:
     [copy] Copying 1 file to C:\dev\addressApp
     [copy] Copying 1 file to C:\dev\addressApp

scaffold-transact-auth-properties:
     [copy] Copying 1 file to C:\dev\addressApp

scaffold-build-number:
     [copy] Copying 1 file to C:\dev\addressApp

scaffold-gitignore:
     [copy] Copying 1 file to C:\dev\addressApp

scaffold:

BUILD SUCCESSFUL
Total time: 0 seconds
```

Your workspace will now contain additional files and folders, including a new build.xml file

Commands

```
ls -l
```

Output

```
$ ls -l
total 53
-rw-r--r-- 1 xxxxxx 1049089    14 Sep 21 08:20 build.number
-rw-r--r-- 1 xxxxxx 1049089  1724 Sep 21 08:20 build.properties
-rw-r--r-- 1 xxxxxx 1049089 14634 Sep 21 08:20 build.xml
-rw-r--r-- 1 xxxxxx 1049089  1311 Sep 21 08:20 new-maestro-lib.properties
-rw-r--r-- 1 xxxxxx 1049089  1377 Sep 21 08:20 new-maestro-project.properties
-rw-r--r-- 1 xxxxxx 1049089  1931 Sep 21 08:20 new-service.properties
drwxr-xr-x 1 xxxxxx 1049089     0 Sep 21 08:20 package/
-rw-r--r-- 1 xxxxxx 1049089   609 Sep 21 08:20 proxy.properties
-rw-r--r-- 1 xxxxxx 1049089 11037 Sep 13 10:11 scaffold.xml
drwxr-xr-x 1 xxxxxx 1049089     0 Sep 21 08:20 src/
-rw-r--r-- 1 xxxxxx 1049089   985 Sep 21 08:20 transact-auth.properties
```

Configure credentials in transact-auth.properties

This file contains properties that identify the TM server, the TM username and password, and the client code (referred to as Organization code in TM).

Edit and save the transact-auth.properties file to provide values for:

- manager.url
- manager.username
- manager.password
- manager.clientCode

Commands

```
vi transact-auth.properties
    or
notepad transact-auth.properties
```

Configure and create a new Application Package

These steps create a new Application Package which will be used to contain and distribute our Maestro form and TM dynamic data service. First we scaffold Application Package properties file, then we edit it to reflect our preferences, and finally we create the new Application Package based on the property values provided.

Issue the Ant command opposite

This will scaffold the file 'new-application-package.properties' so we can edit it in preparation for creating a new Application Package

This file is not created by the default scaffold task in scaffold.xml, so this step is required to create it.

Commands

```
ant -f scaffold.xml scaffold-new-application-package-properties
```

Output

```
$ ant -f scaffold.xml scaffold-new-application-package-properties
Buildfile: C:\dev\addressApp\scaffold.xml

scaffold-new-application-package-properties:
     [copy] Copying 1 file to C:\dev\addressApp

BUILD SUCCESSFUL
Total time: 3 seconds
```

Edit new-application-package.properties

Set the property values from the text opposite

Leave other lines in the file as they are.

Commands

```
vi new-application-package.properties
    or
notepad new-application-package.properties
```

new-application-package.properties

```
application.package.name=CI Application Package
application.package.code=ci-app-package
```

Issue the Ant command opposite

It executes the create-new-application-package task from build.xml to create the folders and files to contain our Application Package

Commands

```
ant create-new-application-package
```

Output

```
$ ant create-new-application-package
Buildfile: C:\dev\addressApp\build.xml

create-new-application-package:
     [echo] Generating Application Pacakge: CI Application Package
[apscaffold] Created new project: C:\dev\addressApp\src\applications\ci-app-package
```

```
BUILD SUCCESSFUL
Total time: 1 second
```

Note that the last Ant task created a new Ant build file specific to our Application Package

It contains tasks such as packaging and deployment.

Commands

```
ls -l
```

```
$ ls -l
total 75
-rw-r--r-- 1 xxxxxx 1049089    14 Sep 20 08:45 build.number
-rw-r--r-- 1 xxxxxx 1049089  1724 Sep 20 08:45 build.properties
-rw-r--r-- 1 xxxxxx 1049089 14634 Sep 20 08:45 build.xml
-rw-r--r-- 1 xxxxxx 1049089  2579 Sep 20 09:32 build-app-ci-app-package.xml
-rw-r--r-- 1 xxxxxx 1049089  1311 Sep 20 08:45 new-maestro-lib.properties
-rw-r--r-- 1 xxxxxx 1049089  1377 Sep 20 08:45 new-maestro-project.properties
-rw-r--r-- 1 xxxxxx 1049089  1928 Sep 20 08:54 new-service.properties
drwxr-xr-x 1 xxxxxx 1049089     0 Sep 20 08:45 package/
-rw-r--r-- 1 xxxxxx 1049089   609 Sep 20 08:45 proxy.properties
-rw-r--r-- 1 xxxxxx 1049089 11037 Sep 13 10:11 scaffold.xml
drwxr-xr-x 1 xxxxxx 1049089     0 Sep 21 11:41 src/
-rw-r--r-- 1 xxxxxx 1049089  1093 Sep 20 09:23 transact-auth.properties
```

Create a new Maestro Form

This step creates our Maestro form. In the form, we create a Form Load rule to call the dynamic data service and populate the street address field. We will be the CI Service in a later step.

Using Maestro, create a new form 'CI Form'

Add a Text Field called 'Street Address'

Add an email field called Email Address

Add a Form Load rule, with the code shown opposite

The code makes a dynamic data call to our service and populates the Street Address field with the response

Maestro Form Load Rule code

```
DynamicData.call("CI Service", {})
.then(function(response) {
        data.streetAddress = response.address.firstLine;
})
```

Download the Maestro form version archive file

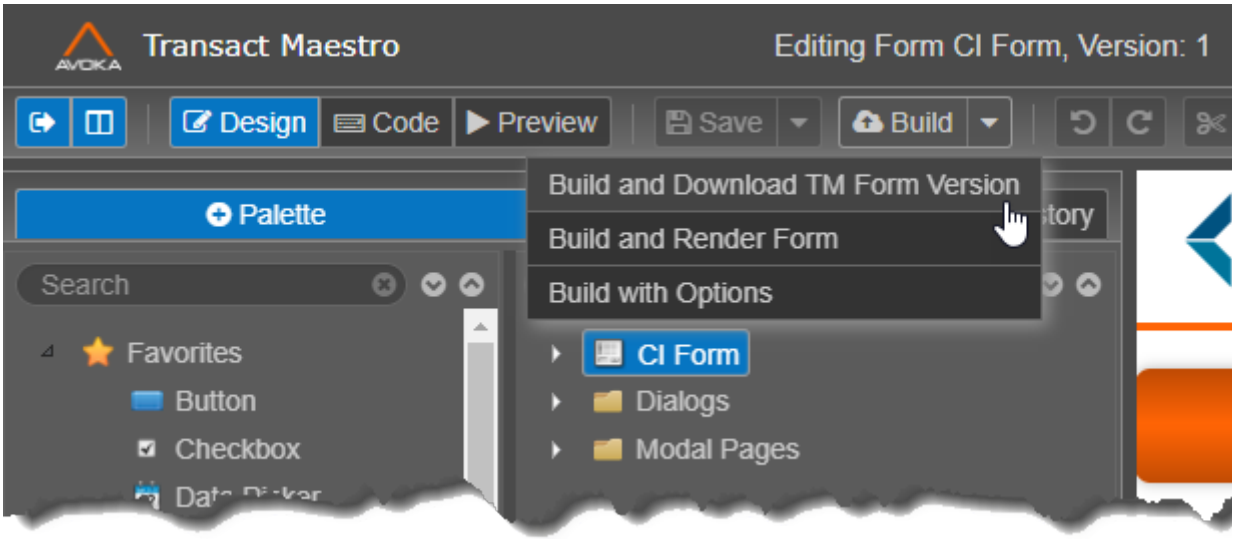
The Maestro form exists only inside our Maestro server at this point in time. We need to download the form's Form Version Archive file from Maestro so it car included in our Application Package.

Download the TM Form Version archive file from Maestro using the Build menu in Maestro

Select 'Build and Download TM Form Version'

The file will be downloaded into your browser

Download folder.



Move the TM Form Version Archive file into the Application Package folder structure

To maintain a neat folder structure, we create a folder under the Application Package forms folder, and move the downloaded TM Form Version Archive file i

Crate the folder structure and move the downloaded Form Version Archive file into it using the commands opposite

(**NOTE:** Change the highlighted filename in the command opposite to reflect the name of the file downloaded from Maestro. You can use tab filename completion to pick the file and folder names for you. Press tab after entering a partial name anywhere to activate completion)

Commands

```
mkdir -p src/applications/ci-app-package/forms/ciform
mv ~/Downloads/form-version-CI_Form-1-2017-09-20.zip src/applications/ci-app-package/forms/ciform
```

Create a form-def.json file describing the TM Form Version

This step creates a form definition file to describe the form version file we just downloaded from Maestro. The form definitiion file includes additional propertie not defined in Maestro, but instead are managed by TM. This form definition file needs to be added to the Application Package definition file in a later step. W create a folder to hold the file in order to have a neater structure.

Create a form-def.json file in the folder to hold the form definition Edit the contents to use the Json shown to the right, replacing the h ighlighted filename with the name of the form version file

Commands

```
touch src/applications/ci-app-package/forms/ciform/ciform-form-def.json

vi src/applications/ci-app-package/forms/ciform/ciform-form-def.json
    or
notepad src/applications/ci-app-package/forms/ciform/ciform-form-def.json
```

ciform-form-def.json

```
{
   "name":"CI Form",
   "formCode":"ciform",
   "currentFormVersionNumber":"1",
```

you downloaded from TM

```
   "formSpaces":[
      {
         "name":"Web Plug-in",
         "anonAccess":true,
         "authAccess":true
      }
   ],
   "formVersions":[
      {
         "versionNumber":"1",
         "formDataEncryption":true,
         "formType":"Maestro Form",
         "formVersionFile":"./form-version-CI_Form-1-2017-09-20.zip"
      }
   ]
}
```

Edit the Application Package definition file to include the form-def.json file

The Application Package definition file contains an array of form-def files included with the package. We need to add our newly-created form-def file to the fo

Edit the Application Package definition file to add the new form-def file to the forms array

Add a new string to the "forms" Json array pointing to the ciform-form-def.json file that you just created

Commands

```
vi src/applications/ci-app-package/app-package-def.json
    or
notepad src/applications/ci-app-package/app-package-def.json
```

app-package-def.json

```
{
  "name": "CI Application Package",
  "description": "Example Application Package for CI development",
  "clientCode": "ci",
  "connections": [],
  "delivery": [],
  "forms": [
    "forms/ciform/ciform-form-def.json"
  ],
  "properties": [],
  "services": []
}
```

Configure and create a new service

Edit the new-service.properties file properties to match the values opposite

Leave other lines in the file as they are

This service will provide dynamic data to the Maestro form when it asks for it in its Form Load rule.

Commands

```
vi new-service.properties
    or
notepad new-service.properties
```

new-service.properties file

```
service.name=CI Service
service.code=ci-service
service.template=Fluent Dynamic Data
service.version=1
```

Issue the Ant command opposite

Commands

```
ant create-new-service
```

Output

```
$ ant create-new-service
Buildfile: C:\dev\addressApp\build.xml

create-new-service:
     [echo] Generating Service: CI Service v1
[gscaffold] Created new project: C:\dev\addressApp\src\ci-service

BUILD SUCCESSFUL
Total time: 4 seconds
```

Note that the last Ant task created a new Ant build file specific to the service

It contains tasks such as deploying and undeploying the service

Commands

```
ls -l
```

Output

```
$ ls -l
total 75
-rw-r--r-- 1 bfrost 1049089    14 Sep 20 08:45 build.number
-rw-r--r-- 1 bfrost 1049089  1724 Sep 20 08:45 build.properties
-rw-r--r-- 1 bfrost 1049089 14634 Sep 20 08:45 build.xml
-rw-r--r-- 1 bfrost 1049089  2579 Sep 20 09:32 build-app-ci-app-package.xml
-rw-r--r-- 1 bfrost 1049089  4979 Sep 20 08:56 build-svc-ci-service.xml
-rw-r--r-- 1 bfrost 1049089   522 Sep 20 09:22 new-application-package.properties
-rw-r--r-- 1 bfrost 1049089  1311 Sep 20 08:45 new-maestro-lib.properties
-rw-r--r-- 1 bfrost 1049089  1377 Sep 20 08:45 new-maestro-project.properties
-rw-r--r-- 1 bfrost 1049089  1928 Sep 20 08:54 new-service.properties
drwxr-xr-x 1 bfrost 1049089     0 Sep 20 08:45 package/
-rw-r--r-- 1 bfrost 1049089   609 Sep 20 08:45 proxy.properties
-rw-r--r-- 1 bfrost 1049089 11037 Sep 13 10:11 scaffold.xml
drwxr-xr-x 1 bfrost 1049089     0 Sep 21 11:41 src/
-rw-r--r-- 1 bfrost 1049089  1093 Sep 20 09:23 transact-auth.properties
```

Move the service into our Application Package folder structure

Our new service was created in a folder structure outside our Application Package folder structure. Although we could still easily reference the newly created situ, we will have a neater structure if we move it into our Application Package folder structure.

Move the new service into the Application Package folder structure and list the folder contents to verify it has been successfully moved

Commands

```
mv src/ci-service src/applications/ci-app-package/services
ls -l src/applications/ci-app-package/services/
```

```
$ ls -l src/applications/ci-app-package/services/
total 4
drwxr-xr-x 1 bfrost 1049089 0 Sep 20 09:49 ci-service/
```

Modify Application Package definition to include the service

Edit the file src/applications/ci-app-package/app-package-def.json

Add a new string to the "services" Json array pointing to the service-def.json file we moved into our folder structure

Commands

```
vi src/applications/ci-app-package/app-package-def.json
    or
notepad src/applications/ci-app-package/app-package-def.json
```

app-package-def.json

```
{
  "name": "CI Application Package",
  "description": "Example Application Package for CI development",
  "clientCode": "ci",
  "connections": [],
  "delivery": [],
  "forms": [
    "forms/ciform/ciform-form-def.json"
```

```
  ],
  "properties": [],
  "services": [
        "services/ci-service/service-def.json"
  ]
}
```

Build the Application Package

Issue the Ant command opposite

This will build the Application Package

Commands

```
ant -f build-app-ci-app-package.xml package-app
```

Output

```
$ ant -f build-app-ci-app-package.xml package-app
Buildfile: C:\dev\addressApp\build-app-ci-app-package.xml

package-app:
[appackage] Created Application Package archive: C:\dev\addressApp\target\application-package-archive
package.zip

BUILD SUCCESSFUL
Total time: 0 seconds
```

Verify our Application Package has been built and saved in the target folder

Commands

```
ls -l target
```

```
$ ls -l target
total 840
-rw-r--r-- 1 bfrost 1049089 857889 Sep 20 12:25 application-package-archive-ci-app-package.zip
```

Deploy the Application Package

Enter the command 'ant -f build-app-ci-app-package.xml upload-app'

This will deploy the Application Package to the server identified in the transact-auth.properties file

Commands

```
ant -f build-app-ci-app-package.xml upload-app
```

Output

```
$ ant -f build-app-ci-app-package.xml upload-app
Buildfile: C:\dev\addressApp\build-app-ci-app-package.xml

upload-app:
 [apupload] Http client: created.
 [apupload] Request: POST http://localhost:9080/manager/secure/rest/application-package/v1/zzzzzzzzz
 [apupload] Response: HTTP/1.1 200 OK
 [apupload] Response: {
 [apupload]   "archiveName": "application-package-archive-ci-app-package.zip",
 [apupload]   "importMessage": "Application Package named 'CI Application Package' for organization
'xxxxxxxxx' was imported successfully.",
 [apupload]   "importStatus": "Completed",
 [apupload]   "importTime": "2017-09-20T12:37+1000"
 [apupload] }

BUILD SUCCESSFUL
Total time: 0 seconds
```

Undeploy the Application Package

To cleanup the server, we can delete the Application Package using 'ant -f build-app-ci-app-package.xml undeploy-app'

(**Note:** An Application Package needs to be undeployed before it can be deployed again)

Commands

```
ant -f build-app-ci-app-package.xml undeploy-app
```

Output

```
$ ant -f build-app-ci-app-package.xml undeploy-app
Buildfile: C:\dev\addressApp\build-app-ci-app-package.xml

undeploy-app:
 [apdelete] Http client: created.
 [apdelete] Request: DELETE http://localhost:9080/manager/secure/rest/application-package/v1/xxxxxxxx
/CI+Application+Package HTTP/1.1
 [apdelete] Application Package Deleted: xxxxxxxxx/CI Application Package

BUILD SUCCESSFUL
Total time: 0 seconds
```

Next Steps

Source Control

Our entire project template can now be committed to source control. Since your choice of source control system is up to you, we do not provide any specific instructions here, but on the assumption that Git is popular, the original scaffold task already created a .gitignore file for us. The .gitignore file is configured to ***not*** put transact-auth.properties into source control. If you are using Git, you can safely check your project into the Git project you will be working under. The following pages also include a small tutorial for working with GitLab.

CI Server Automation

At this stage, we have accomplished the goal of creating and deploying a new service and form, and the associated assets are all sitting in source control. So far, we have done all of this work manually, as the scenario called for a new development effort.

We can now start to introduce automation to trigger a CI process. Any changes made to our project assets are committed to source control. We can use the commit as a trigger to initiate a CI workflow, or we can elect to manually trigger from the CI server console.

The first step of a CI workflow is to gather the required assets from source control to prepare for a build. The next step is the build, which will package and deploy our Application Package to chosen servers.

Since some of our assets live in Maestro, changes made to the Maestro form must be manually downloaded and committed to source control.

Now we can go about creating an automated process on a CI server such as Jenkins, which will:

1. Detect changes in source control
2. Extract content from source control to a build area
3. Build the Application Package
4. Deploy the Application Package to the chosen TM server

Since your choice of automation environment is entirely up to you, we don't provide any specific instructions here, but the following pages do contain a small tutorial for configuring Jenkins.

Jenkins Tutorial

Here you can follow a brief tutorial to install the Transact Fluent SDK, configure a working Jenkins CI server, then scaffold, type-check, package, upload, and remotely test a service on your Transact Manager server.

Prerequisites

This tutorial assumes you already have these prerequisites in place:

- You have downloaded the latest Transact Fluent SDK - you can download from the download page Transact Fluent SDK Download
- You have downloaded and installed a Java ***JDK*** - download version 8 from http://www.oracle.com/technetwork/java/javase/downloads/index.html
- You have downloaded and installed Apache Ant - download latest from http://ant.apache.org/bindownload.cgi
- You have downloaded Jenkins - https://jenkins.io/download/ - using the Generic Java package (.war) - instructions in this exercise use this WAR package

In a subsequent tutorial, we'll show how to hook Jenkins to a version control system so you can start to build a CI pipeline from this simple beginning.

Recipe

We will be performing the following steps:

1. Create a folder structure
2. Install the Transact Fluent SDK
3. Install Jenkins WAR distribution
4. Run Jenkins
5. Create a Jenkins admin user
6. Configure Jenkins global settings
7. Create a Jenkins job
8. Trigger the Jenkins job
9. View the Jenkins job results

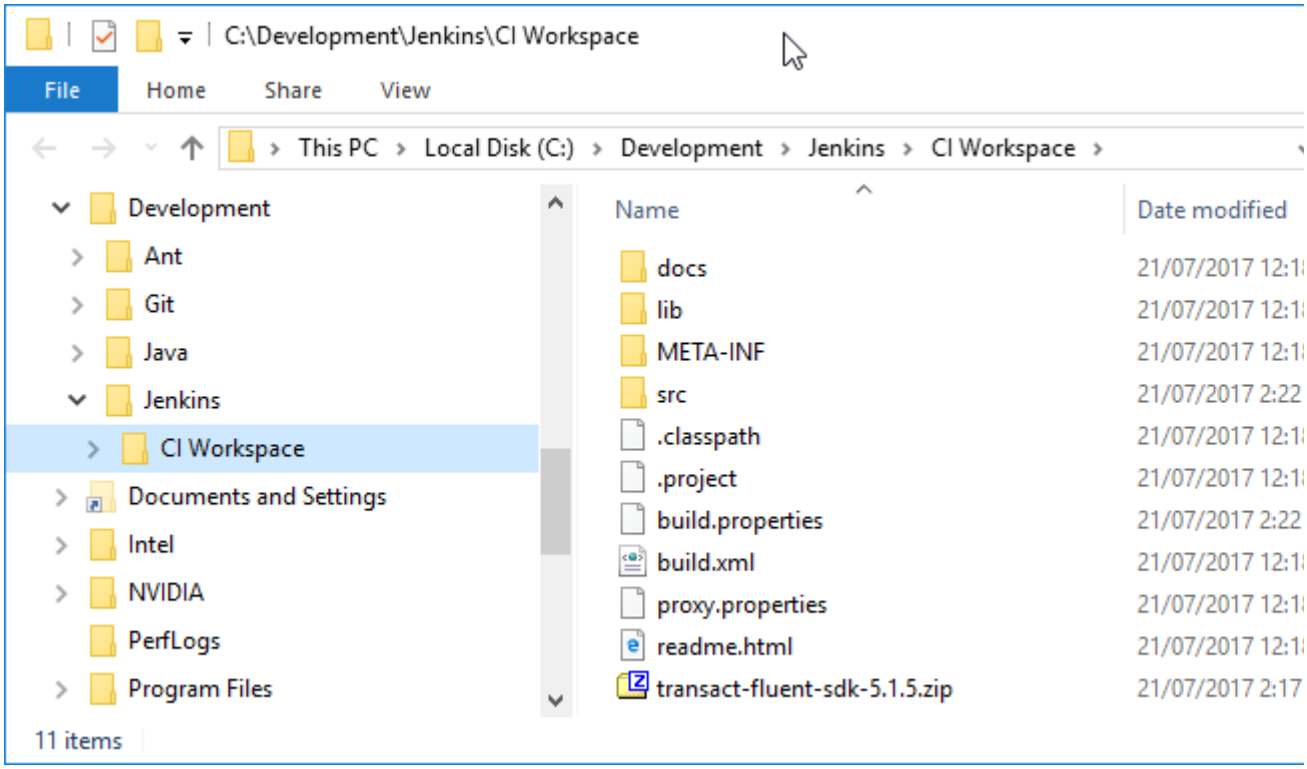
Step-by-step Instructions

Step	Screenshot/Details

Create the folder structure by opening a Windows CMD prompt and entering the commands:
 	
(Windows-key then type CMD to find it)
	`mkdir C:\Development\`
	`mkdir C:\Development\Ant`
	`mkdir C:\Development\Git`
	`mkdir C:\Development\Java`
	`mkdir C:\Development\Jenkins`
	`mkdir C:\Development\Jenkins\CI Workspace`

Unzip the Transact Fluent SDK into the CI Workspace folder Your folder structure should	

look like this

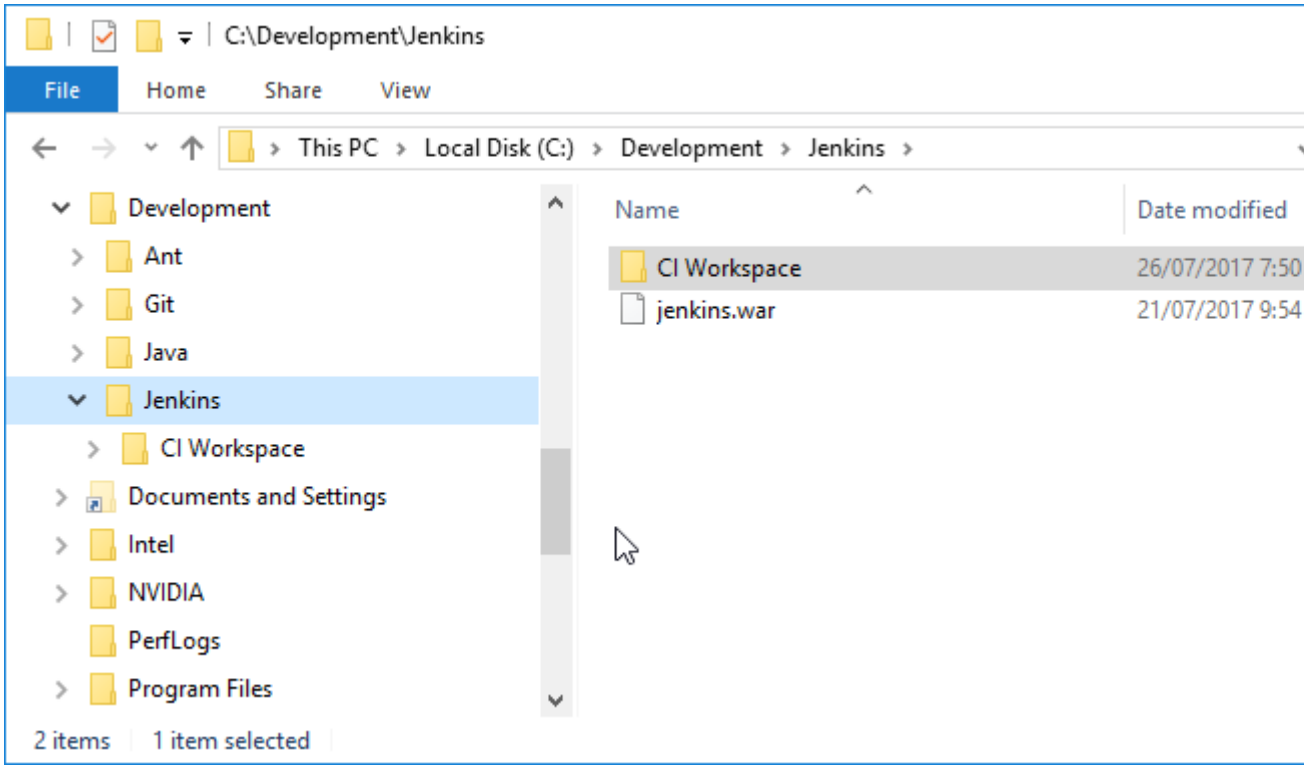


Edit the build.properties file to change username, password and client code details

```
# TM Environment Specific Properties

# Generally you would not check this file into source control

# TM Server URL
manager.url=http://192.168.20.133:9080/manager/

# TM REST User Login Name
manager.username=administrator

# TM REST User Password
manager.password=

# TM Organization Client Code
manager.clientCode=
```

Copy jenkins.war into C:\Development\Jenkins

Your folder structure should look like this

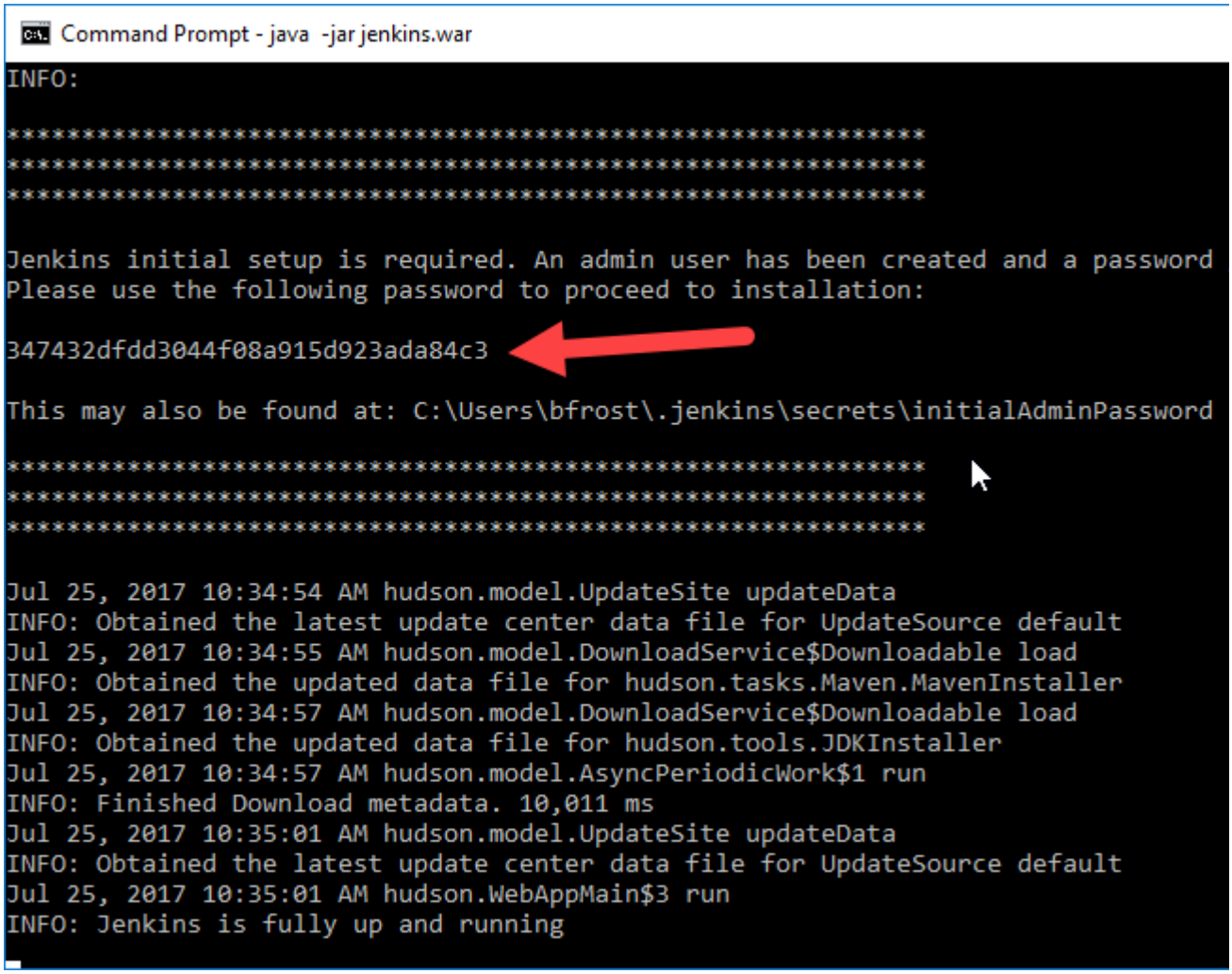


Start Jenkins from the CMD prompt with the commands:

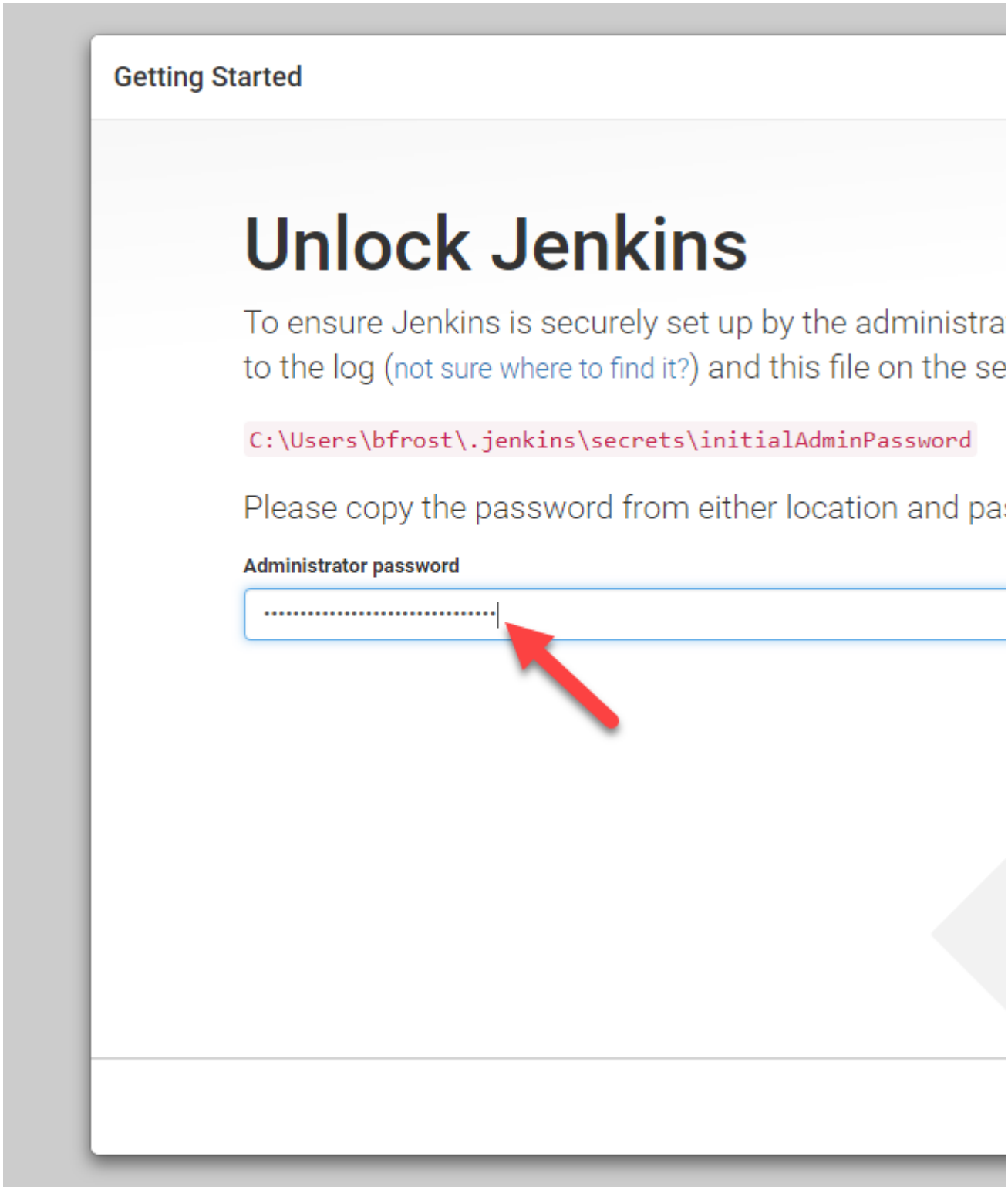
```
cd C:\Development\Jenkins
java -jar jenkins.jar
```

After a few seconds, you will see a one-time password

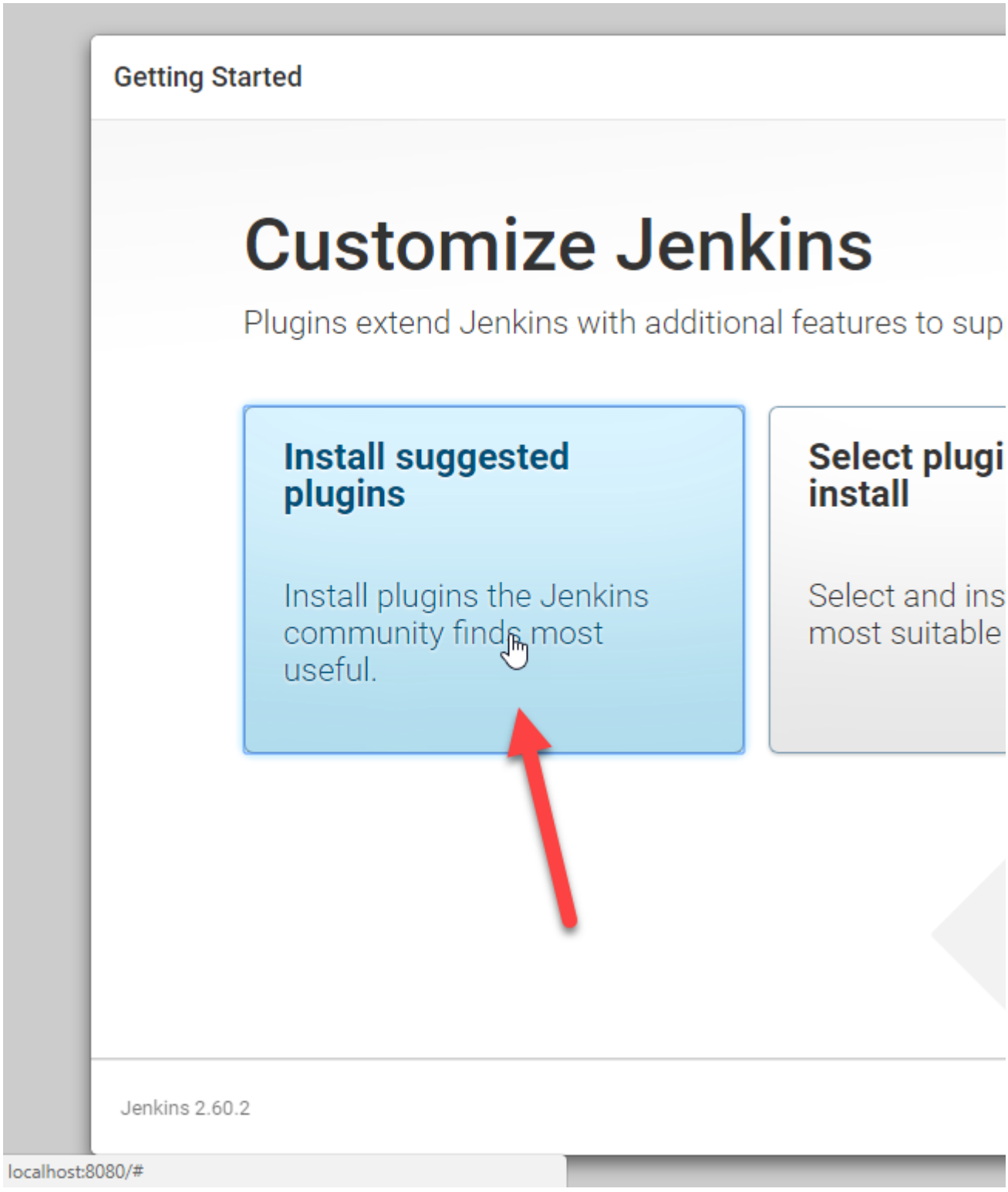
Copy the password into the clipboard



Open a browser at the URL:	http://localhost:8080
Paste the password and click Continue	

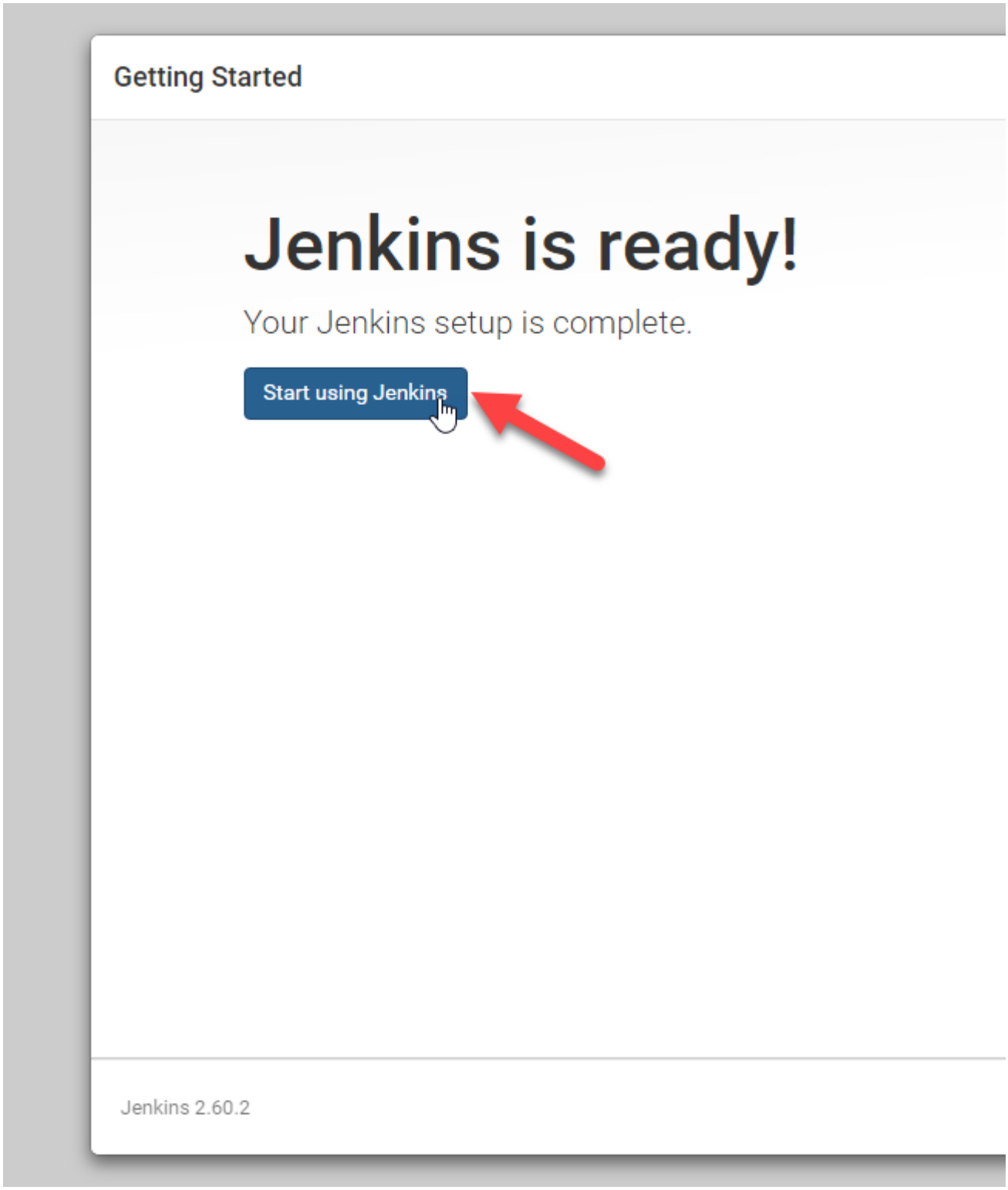


Click Install suggested plugins



After a few minutes, Jenkins is ready

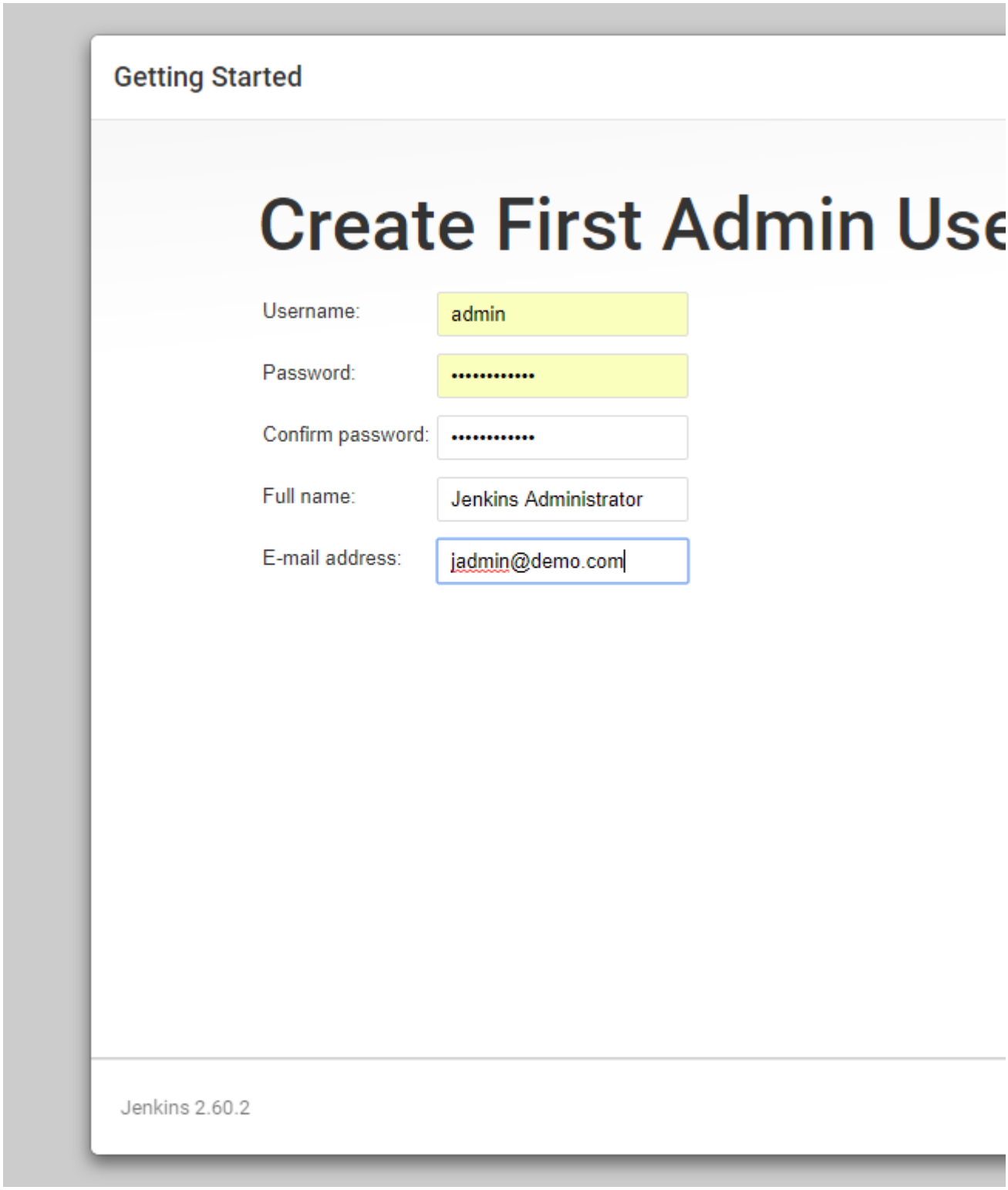
Click Start using Jenkins



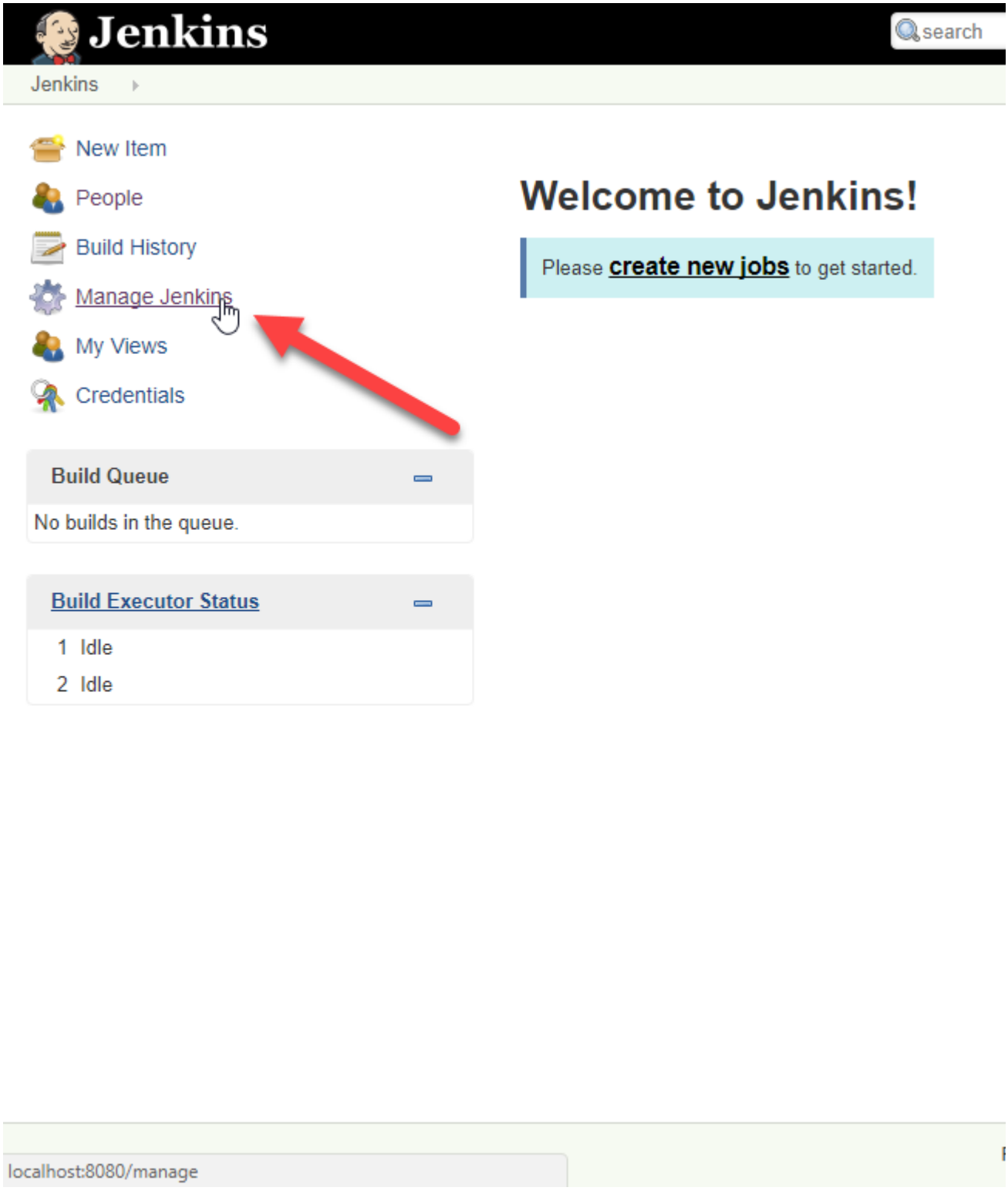
Enter details of the admin user

Click Save and Finish

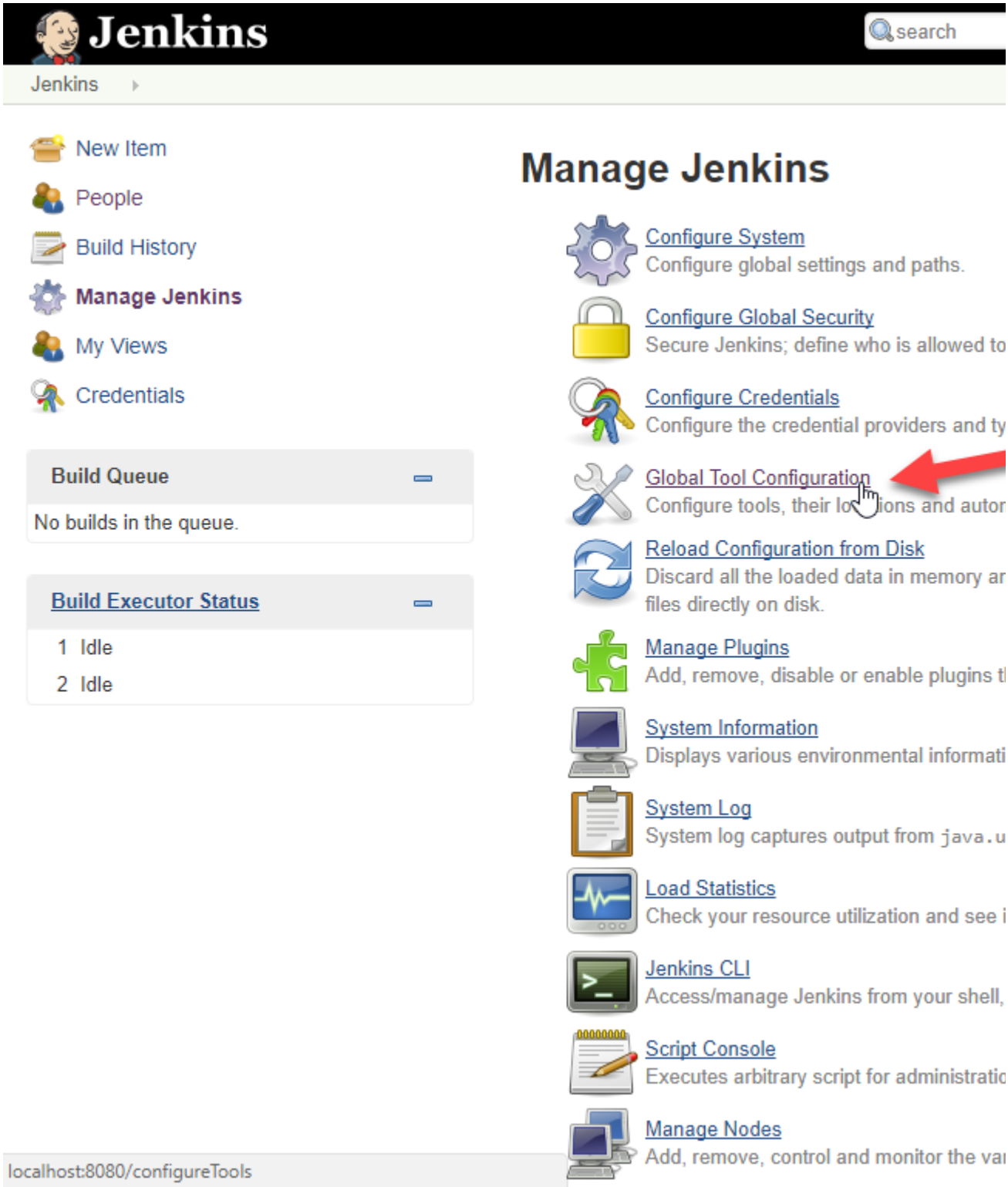
(DO NOT click Continue as admin)



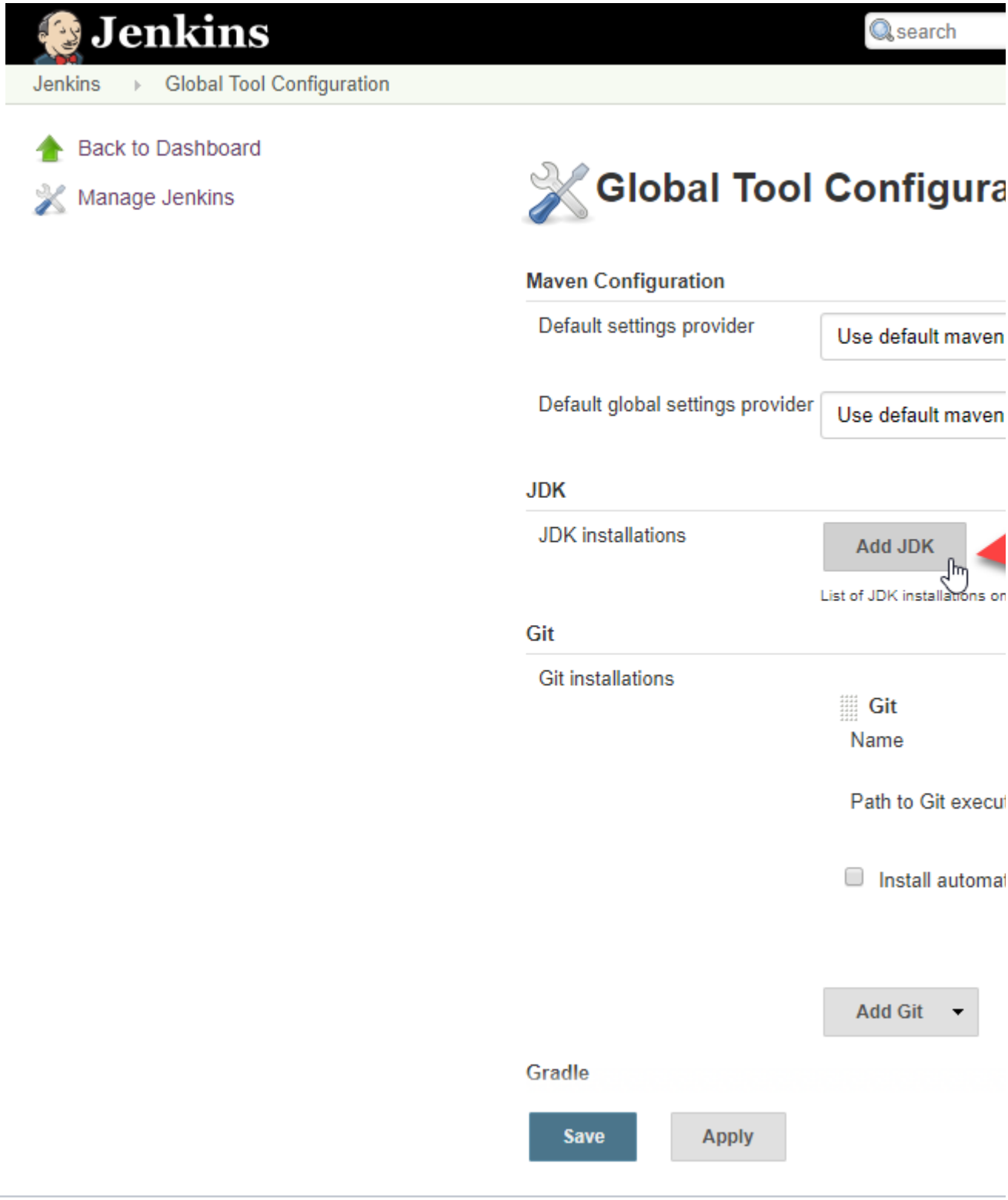
Click Manage Jenkins



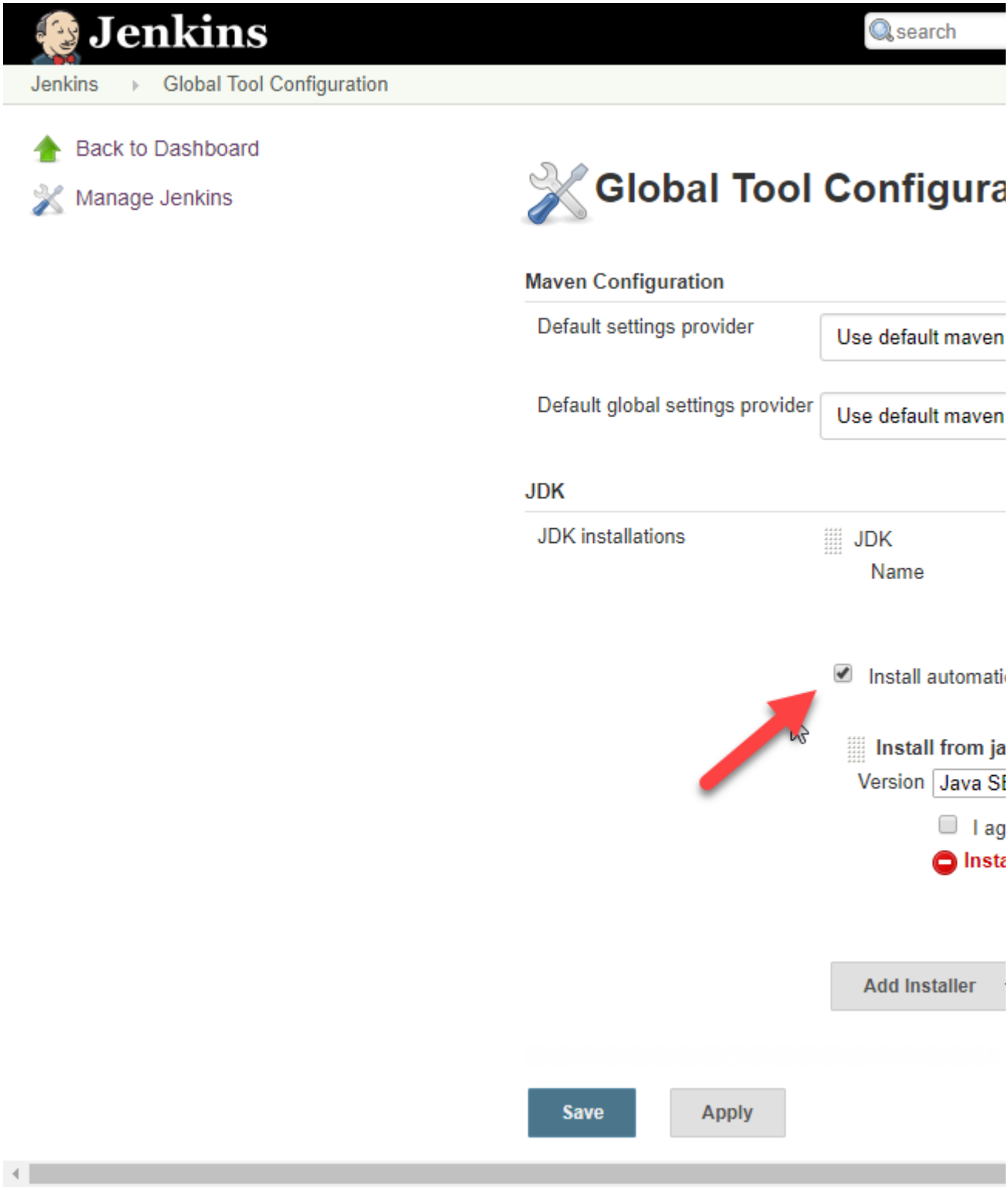
Click Golbal Tool Config uration



Click Add JDK

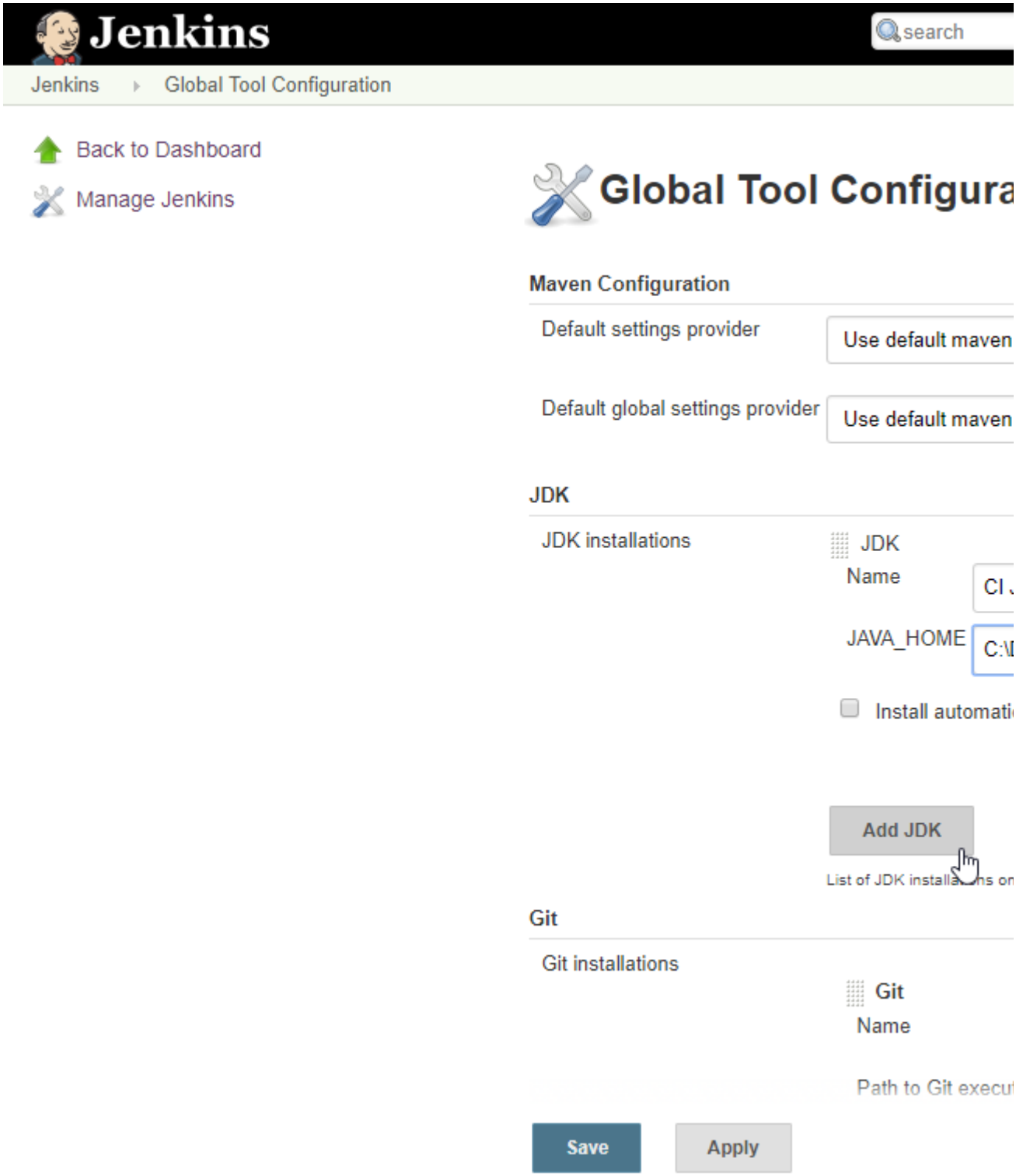


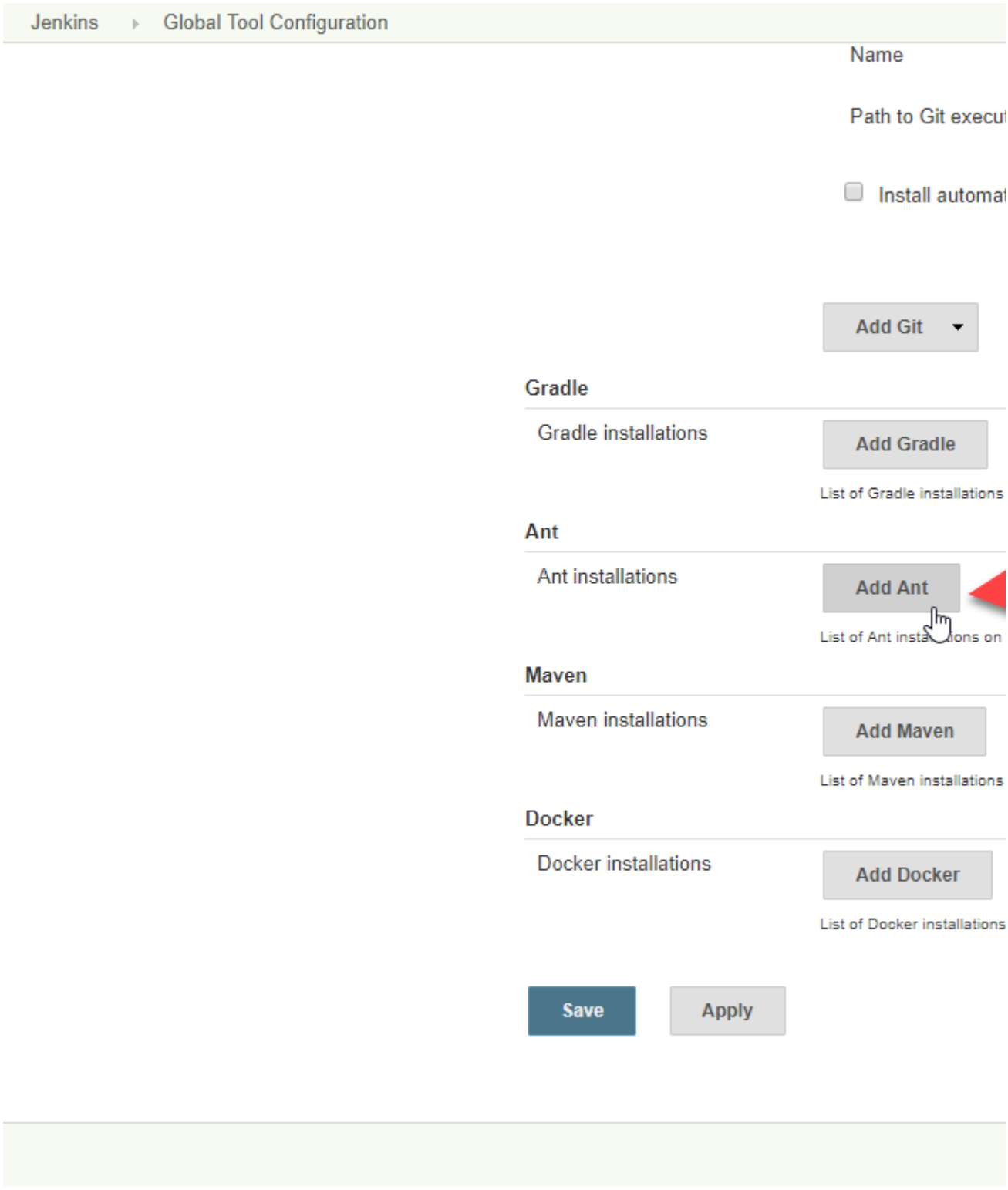
Uncheck Install automatically



Enter a Name of your choice

Enter the path to where the JDK was unzipped in JAVA_HOME

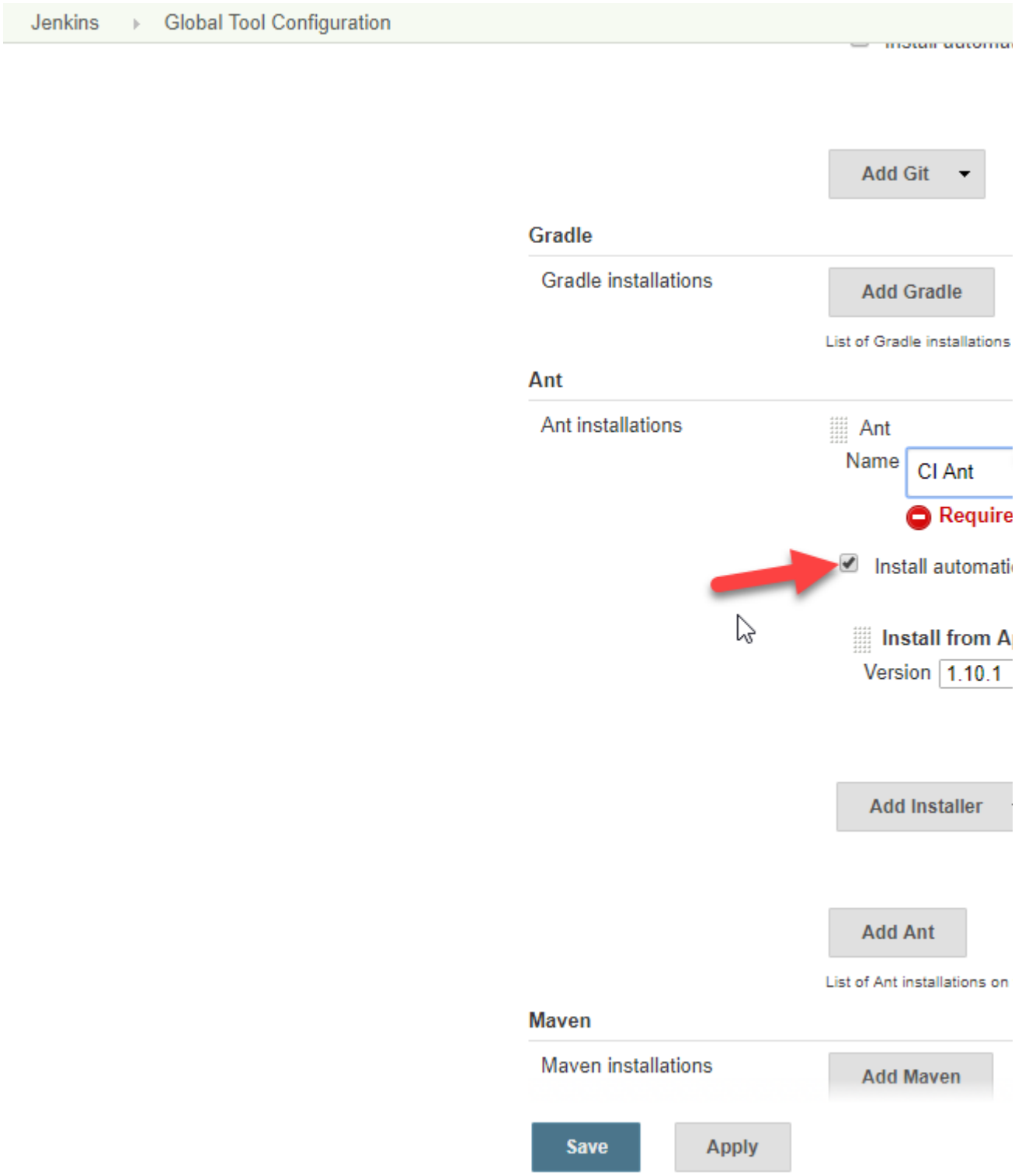

Jenkins
search
Jenkins
Global Tool Configuration
Back to Dashboard
Manage Jenkins
Global Tool Configura
Maven Configuration
Default settings provider
Use default maven
Default global settings provider
Use default maven
JDK
JDK installations
JDK
Name
JAVA_HOME
Install automati
Add JDK
Git
Git installations
Git
Name
Path to Git execut
Save
Apply



Scroll down to Ant

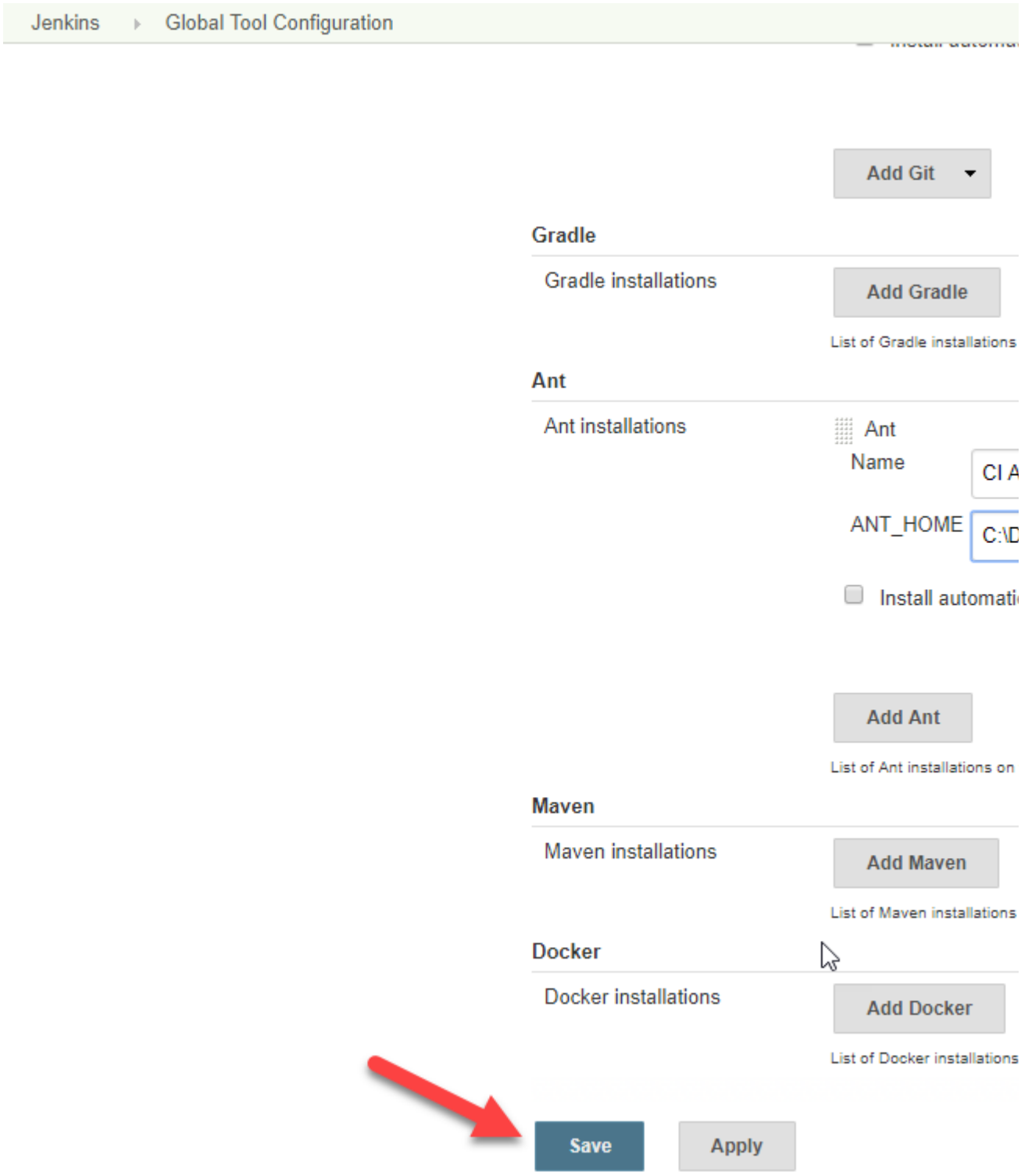
Enter a Name of your choice

Uncheck Install automatically



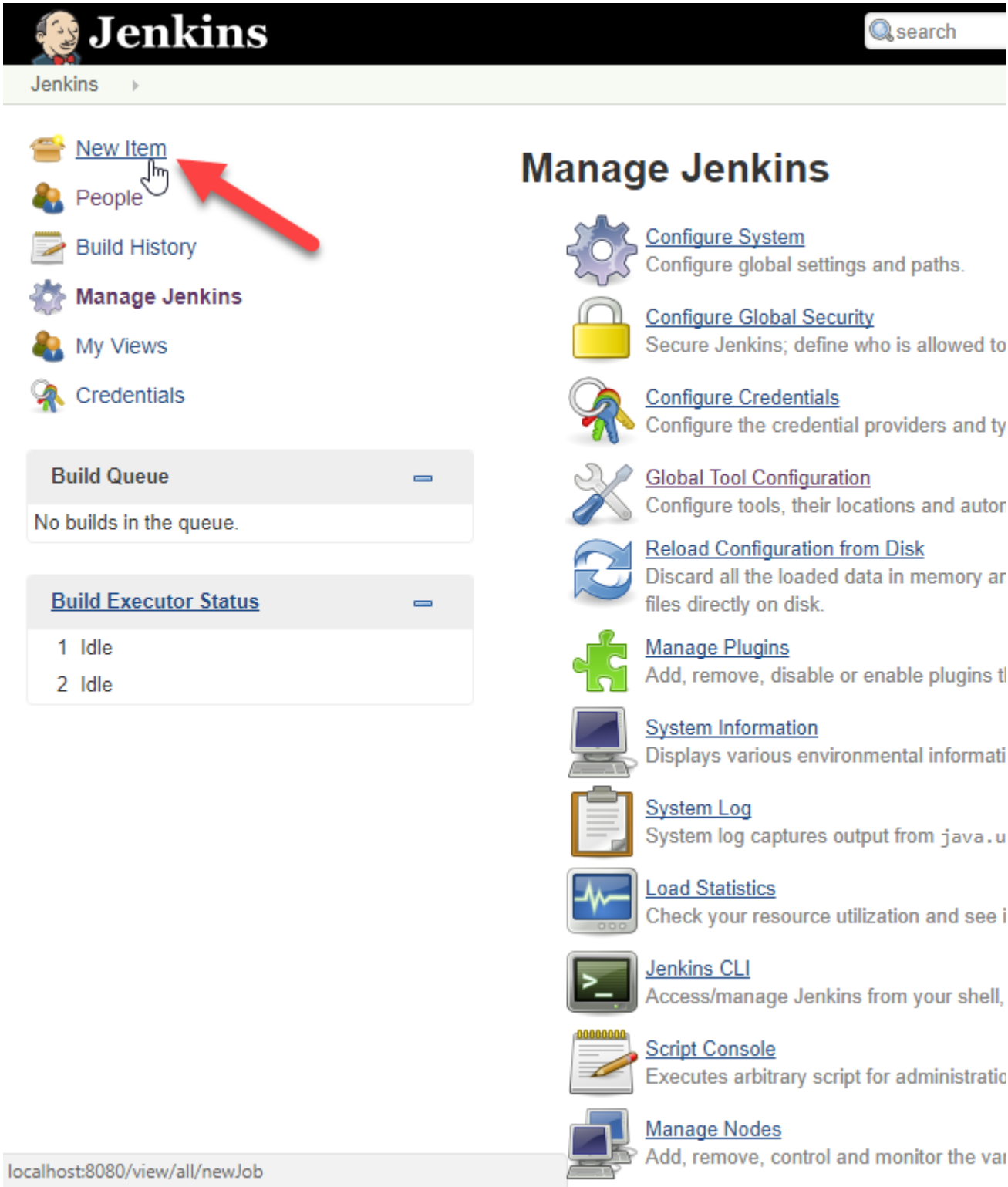
Enter the path to the Ant installation in ANT_HOME

Click Save



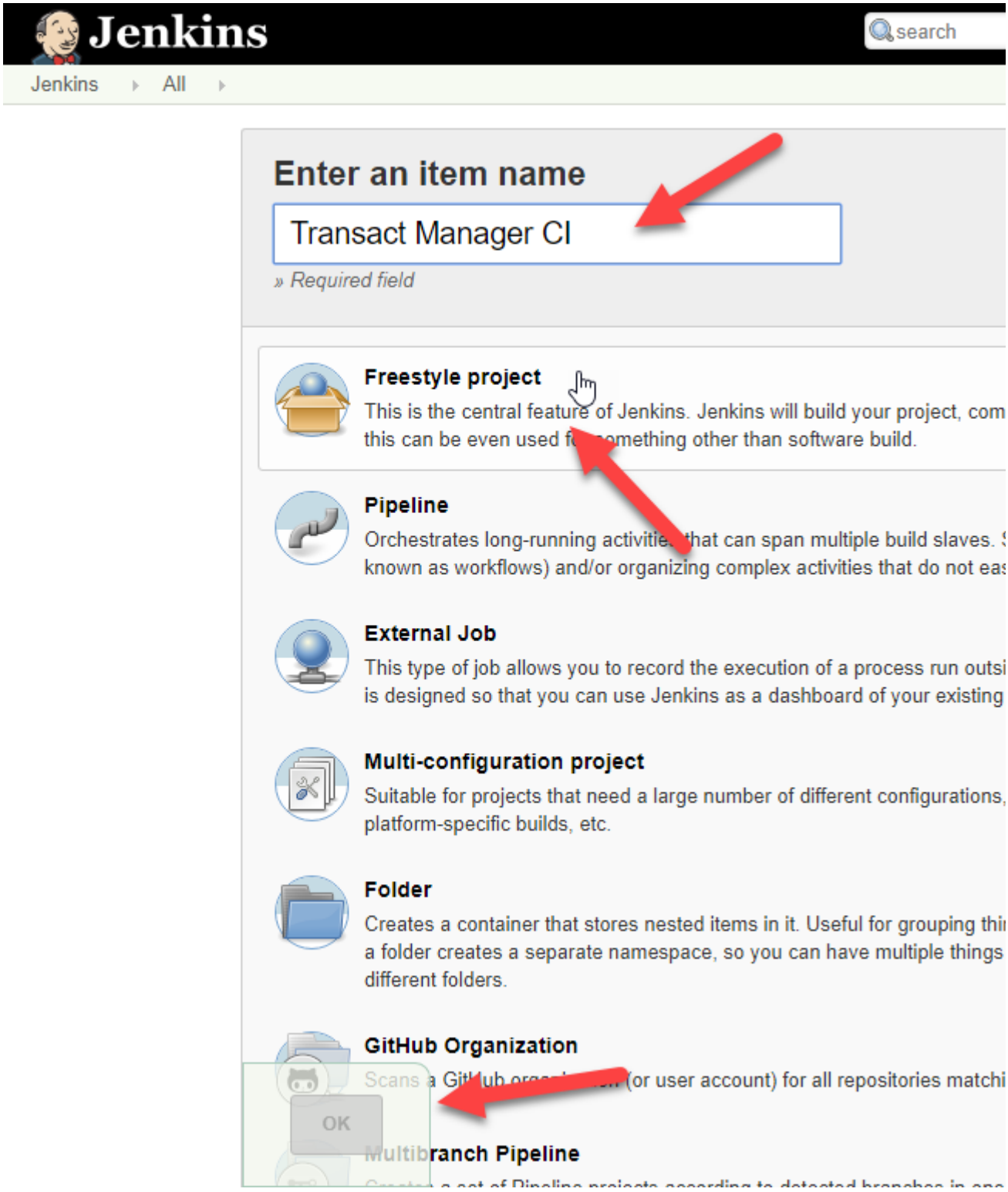
Now Jenkins has been configured with a JDK and an Ant installation, a new CI project can be created

Click New Item

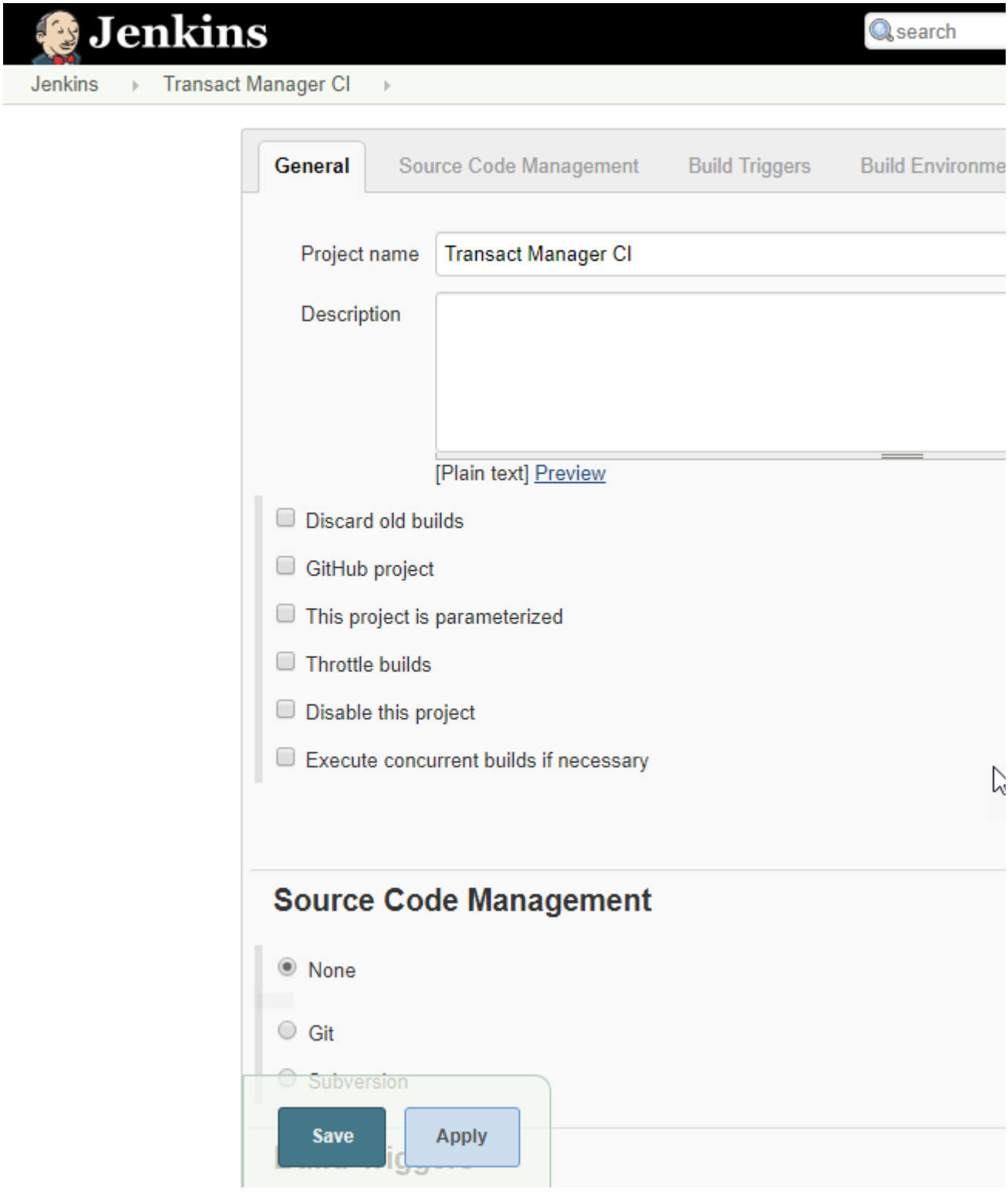


Enter a name for the project

Click Freestyle project

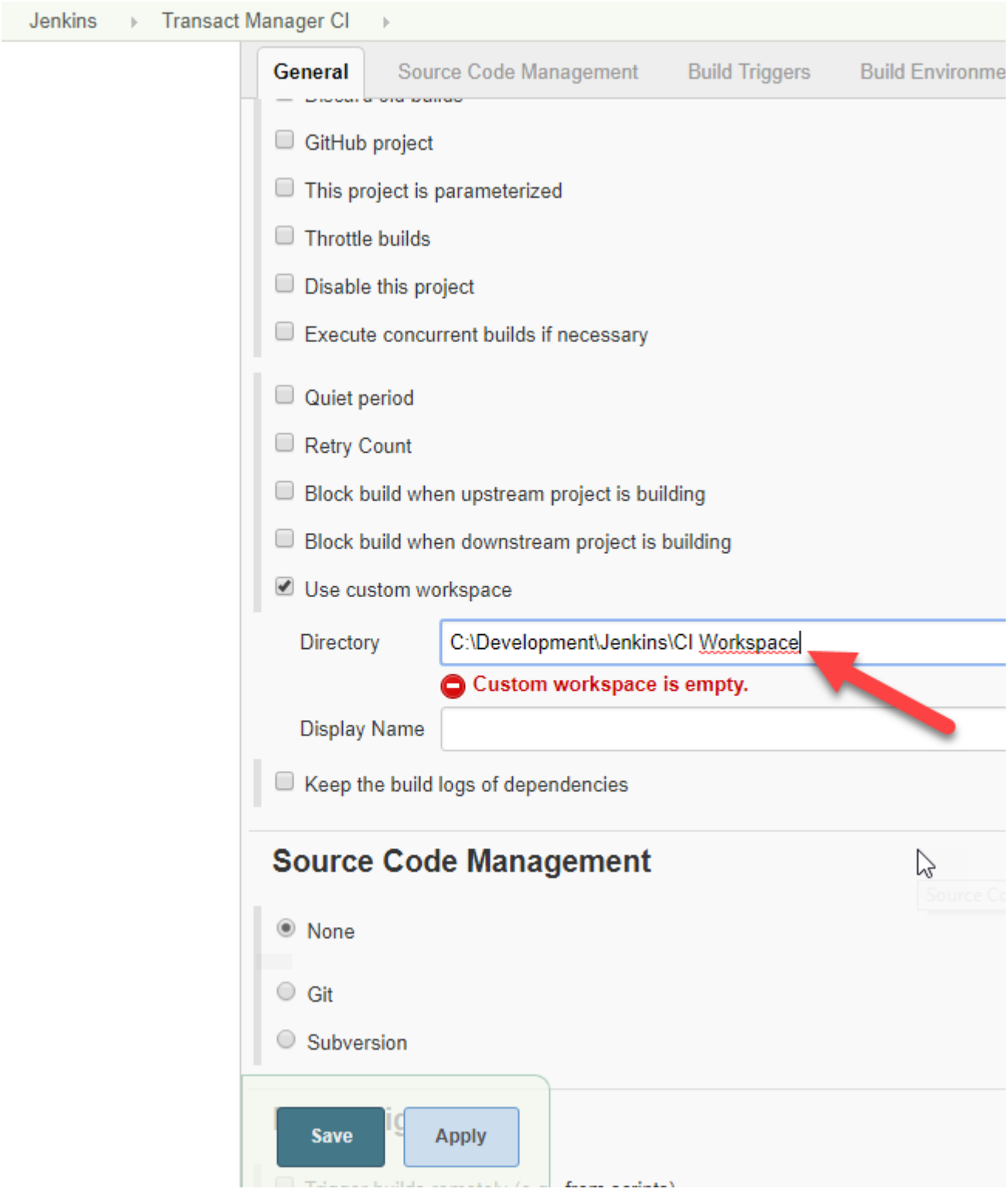


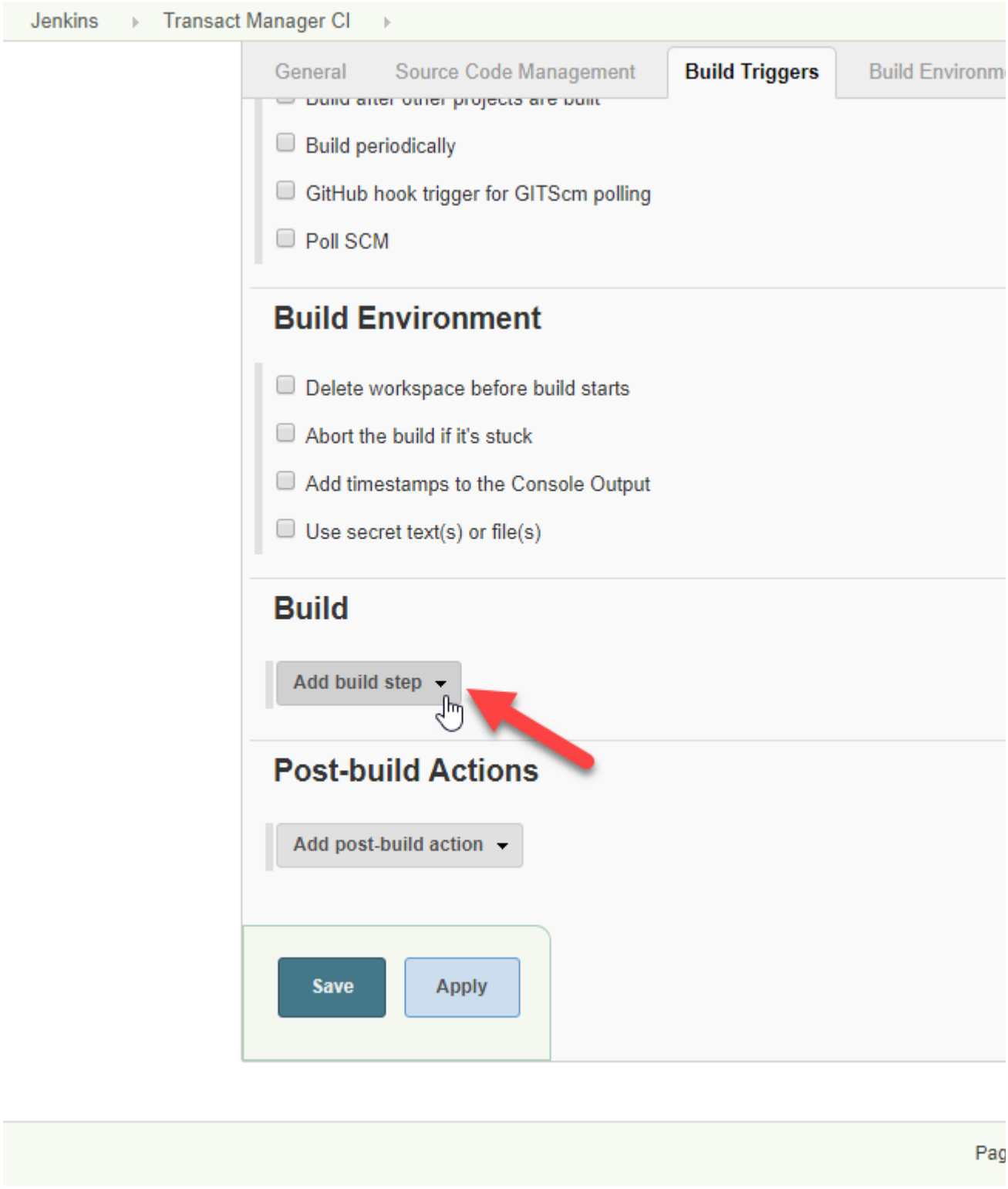
Click the Advanced... button



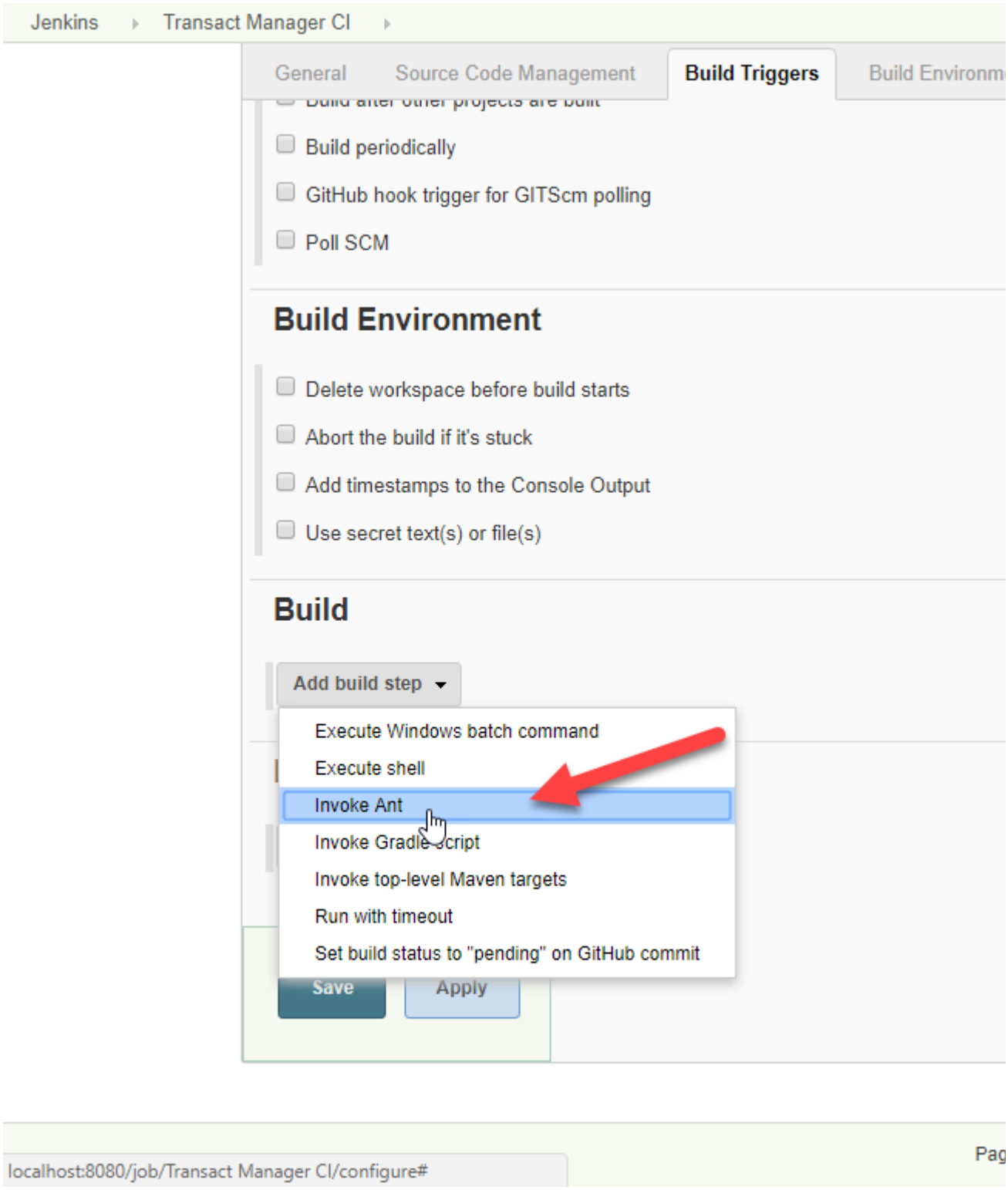
Click Use custom workspace

Enter the path where you will be storing the Transact Fluent SDK and project

source files	
Scroll down to Build Click Add build step	



Select the Invoke Ant step

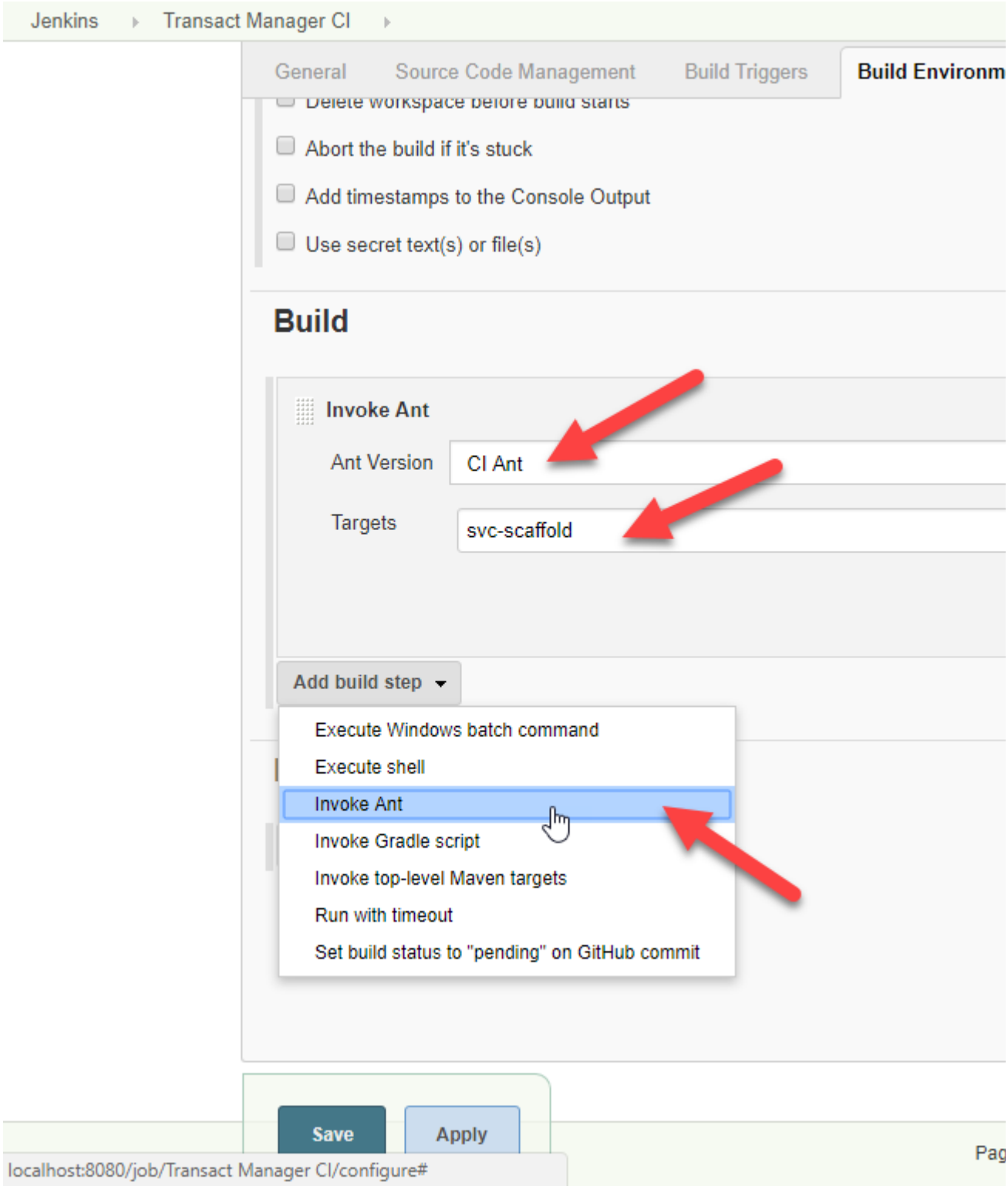


Select the Ant Version configured earlier

enter svc-scaffold in the Targets field

Add another Invoke

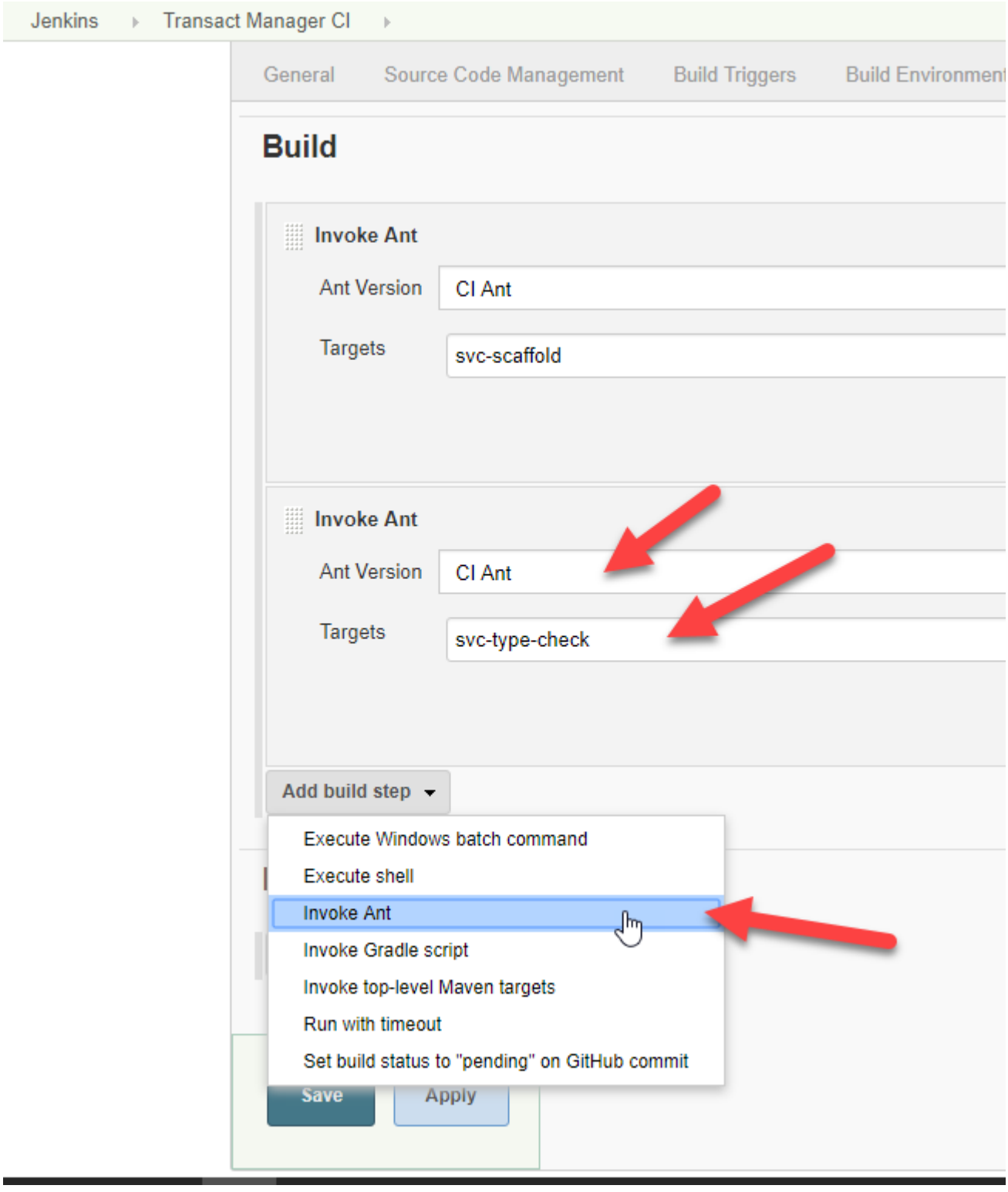
Ant build step



Select the Ant Version

Enter svc-type-check in Targets

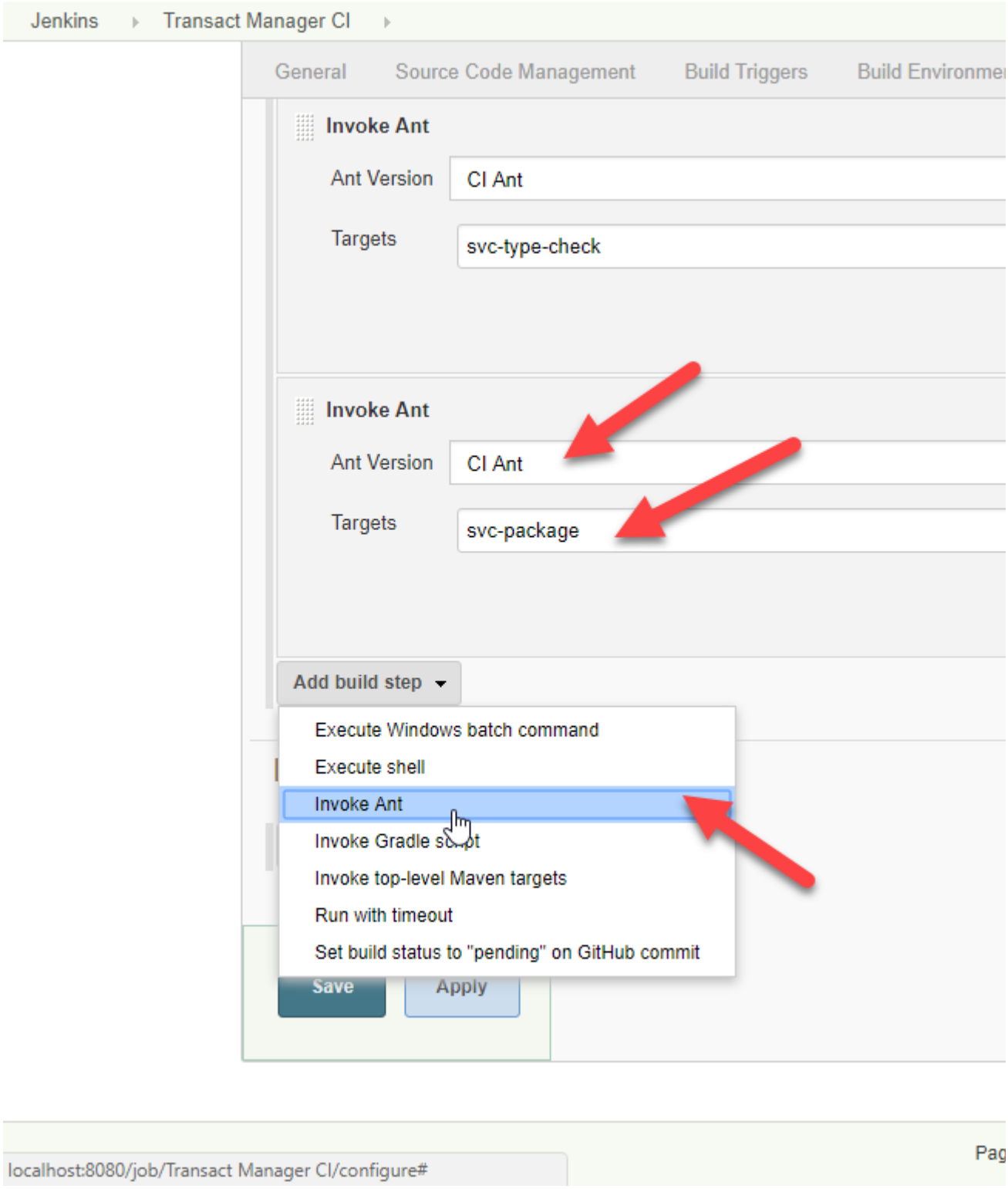
Add another Invoke Ant build step



Select the Ant Version

Enter svc-package in Targets

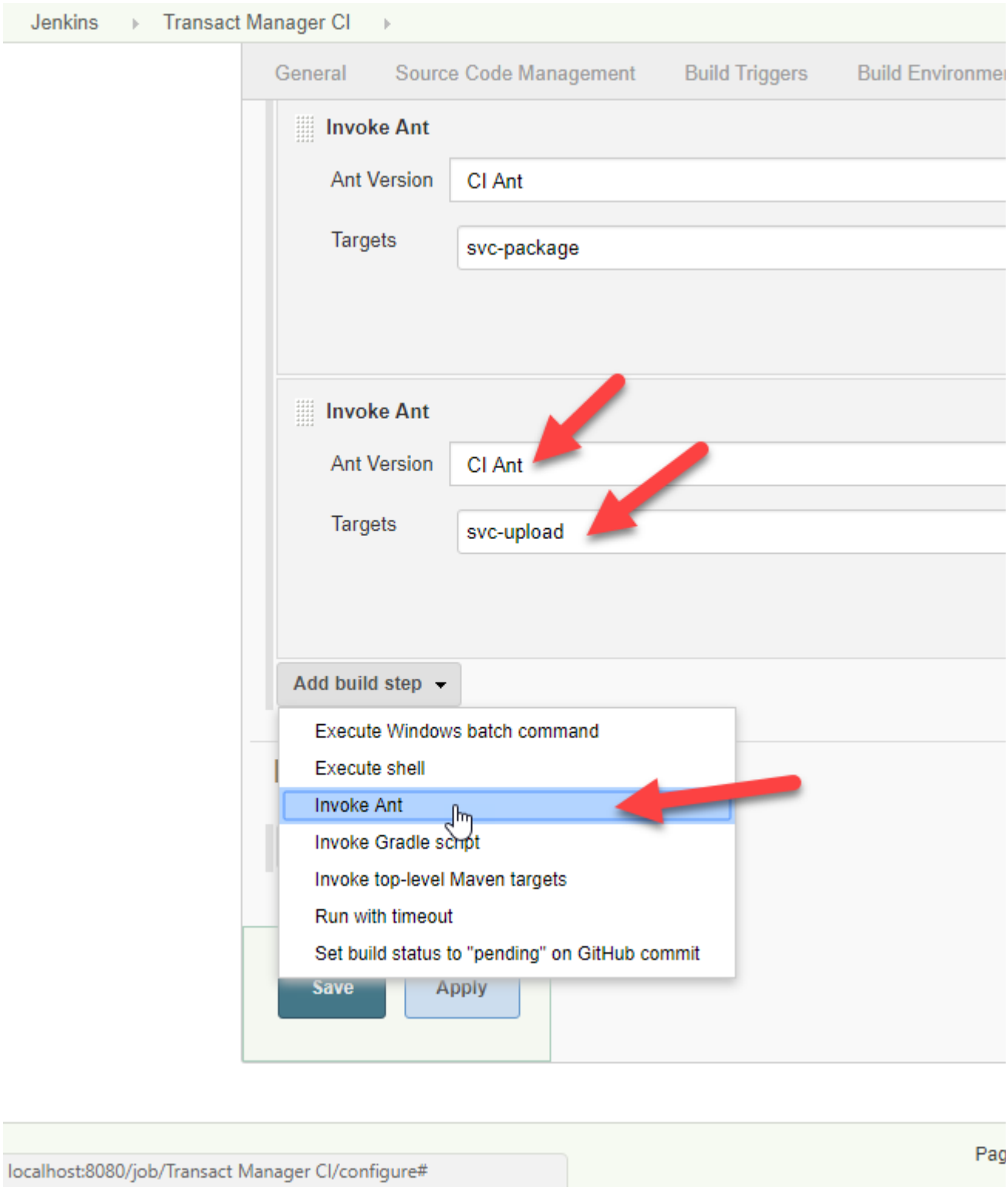
Add another Invoke Ant build step



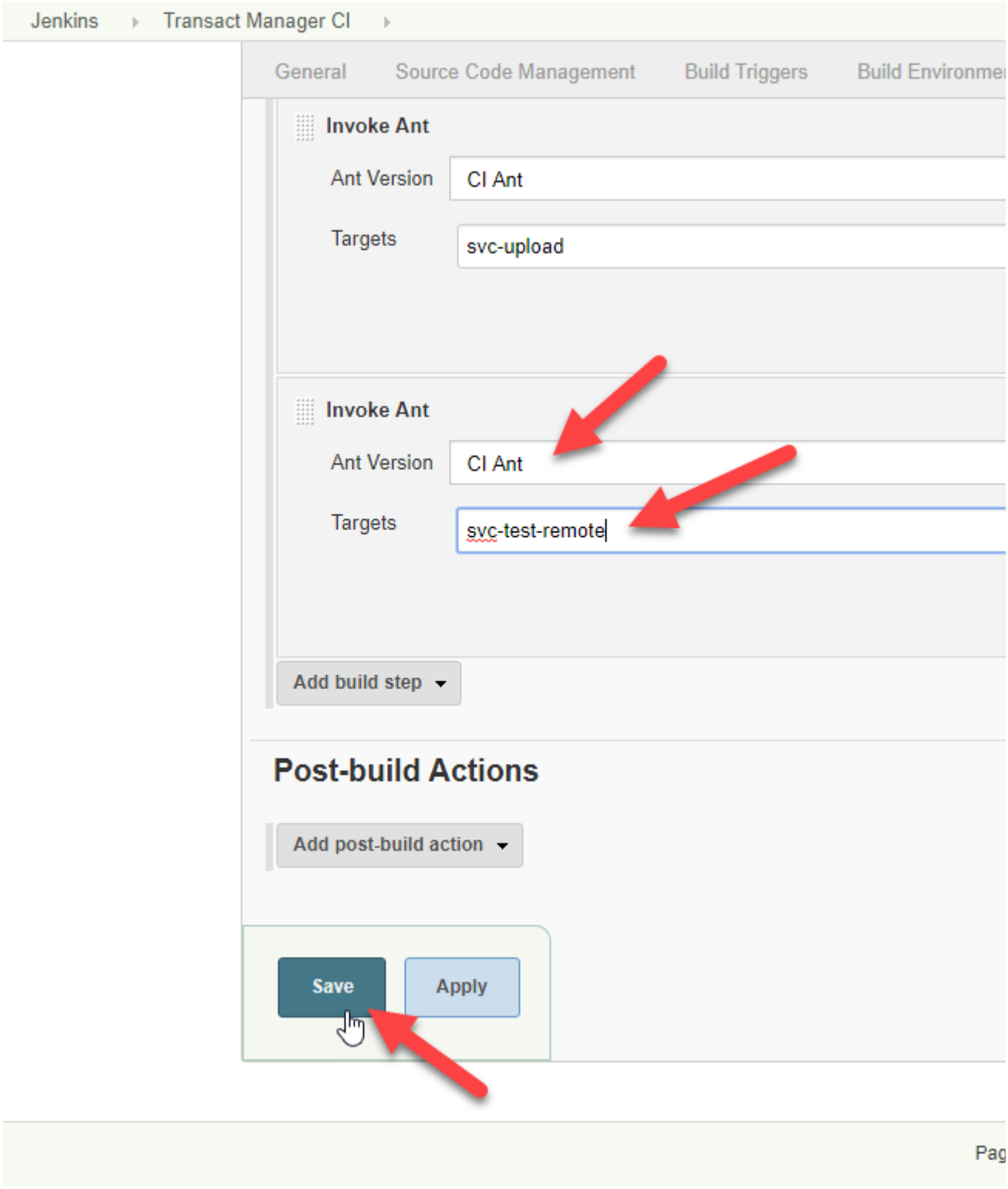
Select Ant Version

Enter svc-upload in Targets

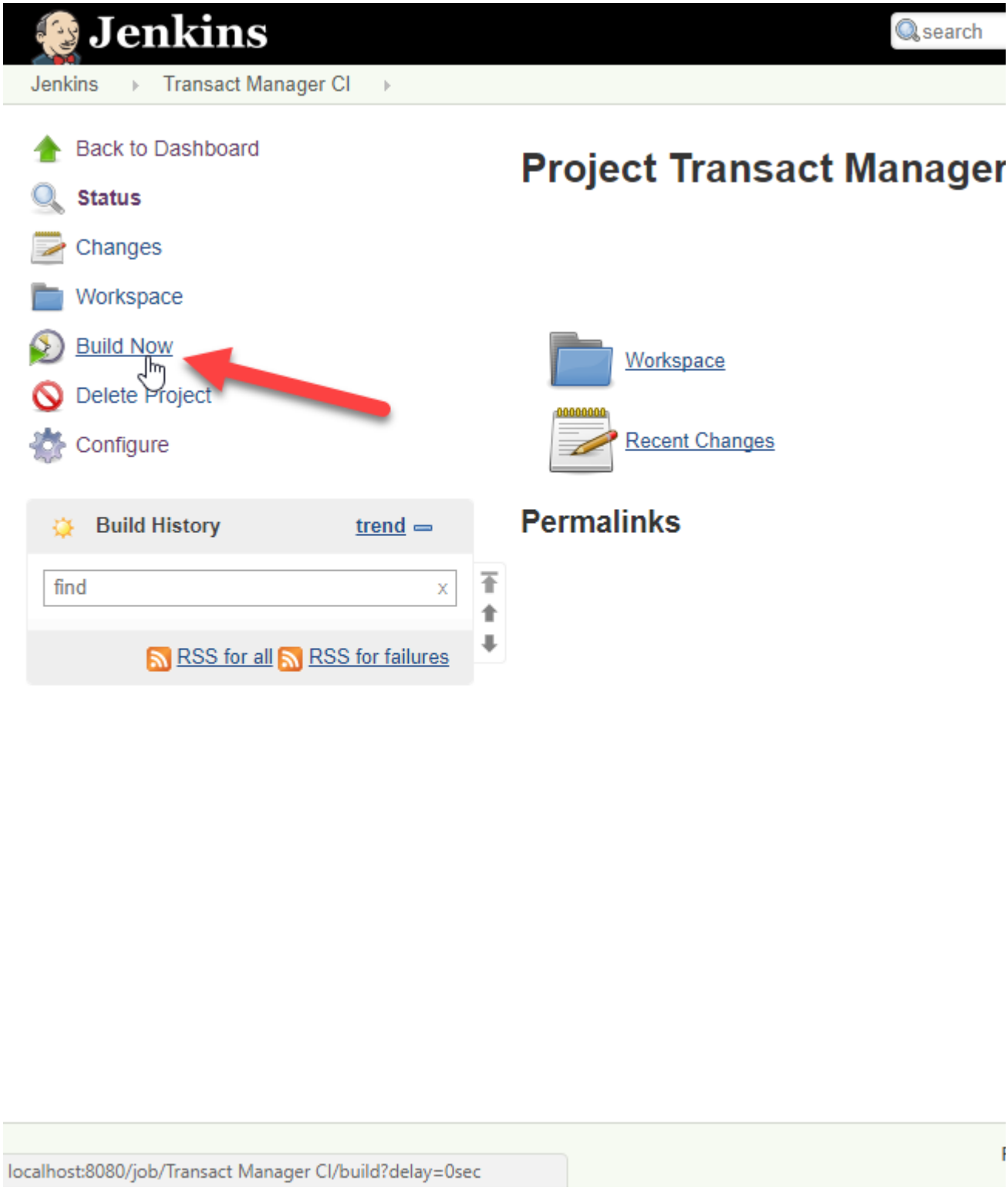
Add another Invoke Ant build step



Select Ant Version Enter svc-test-remote in Targets Add another Invoke Ant build step Click Save	

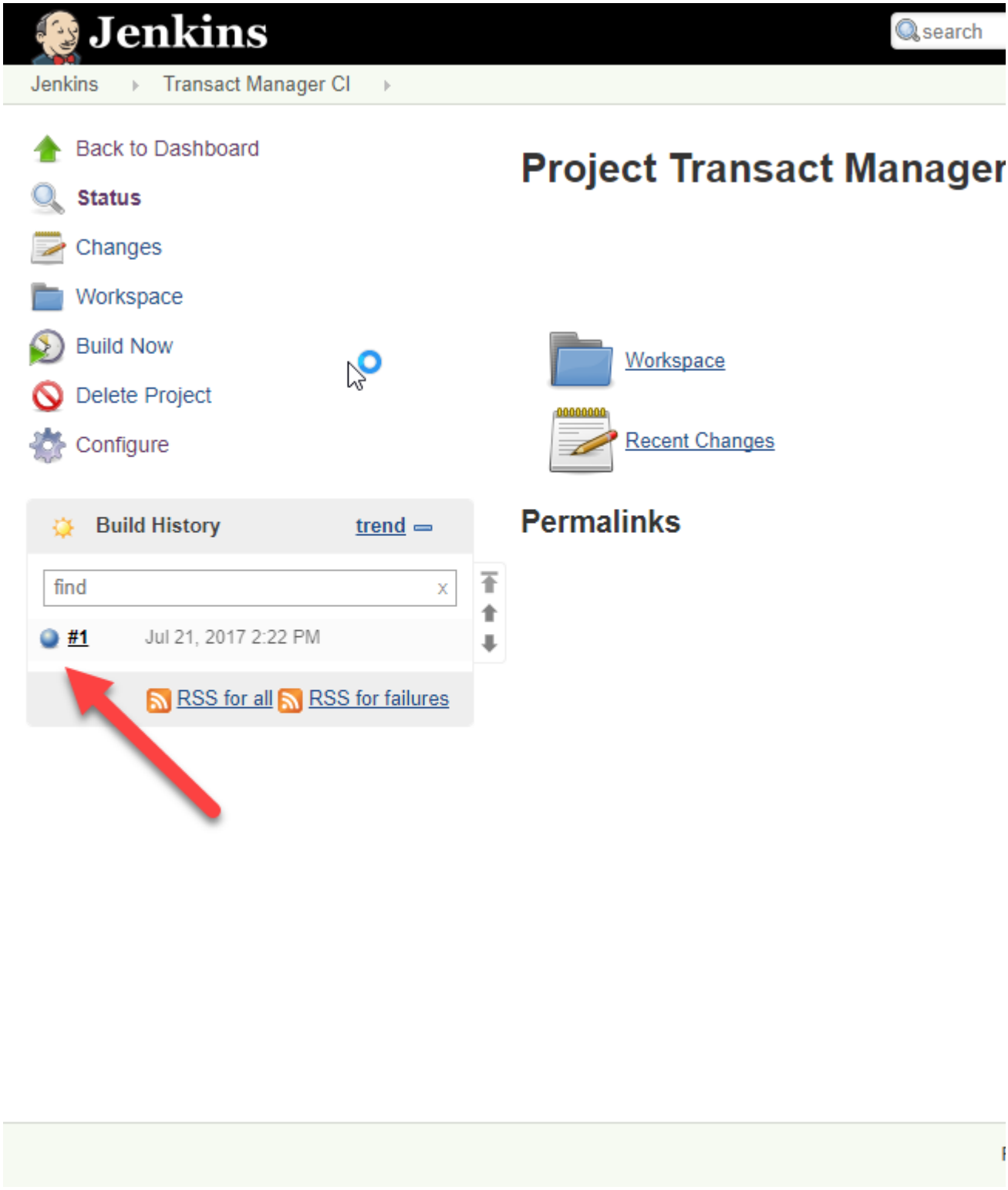


Click Build Now

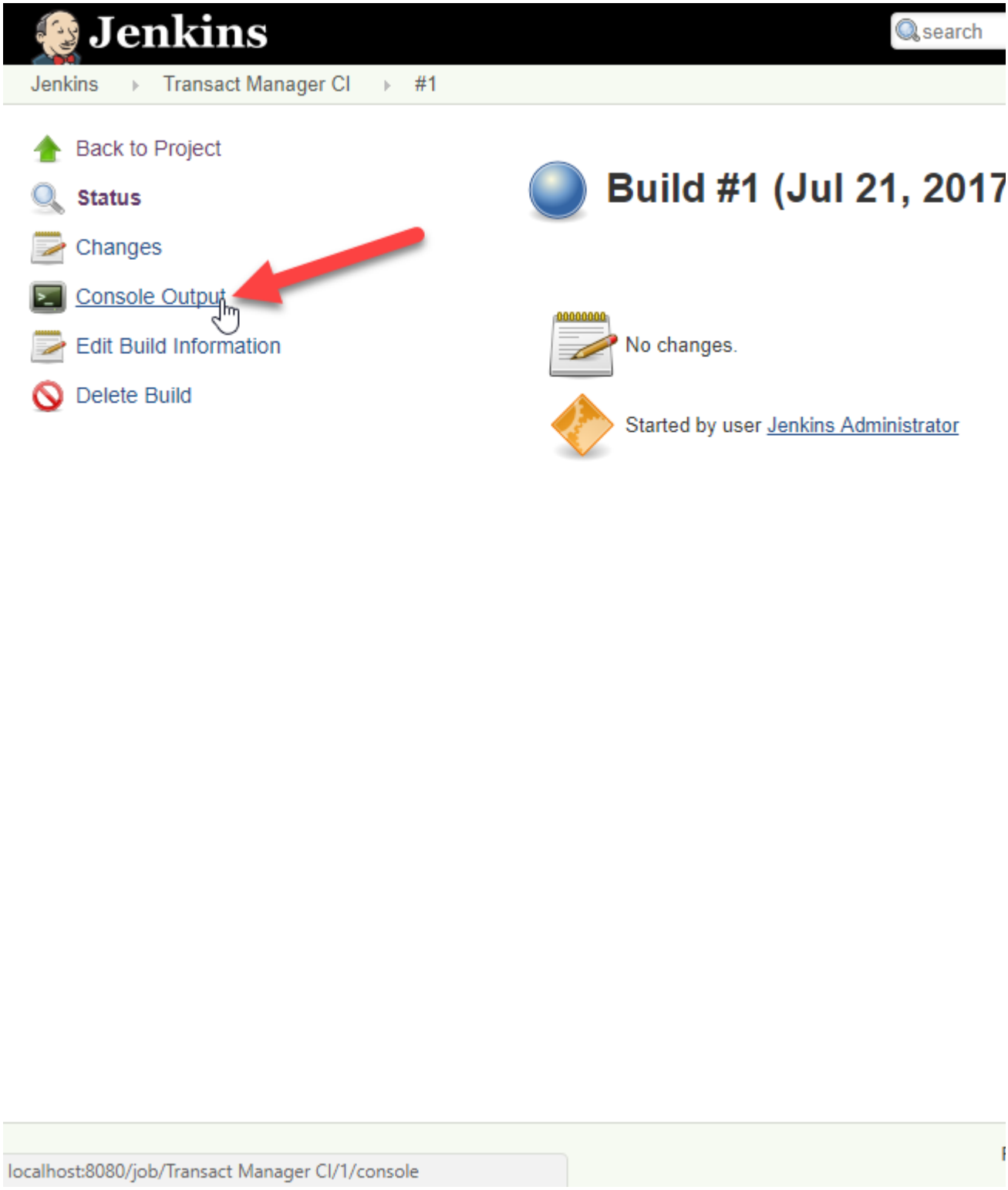


The project will run briefly

Click the build number link in Build History



Click Console Output



You can see the output from each Ant task

Jenkins ▸ Transact Manager CI ▸ #1

- Back to Project
- Status
- Changes
- **Console Output**
  - View as plain text
- Edit Build Information
- Delete Build

Executed Ant Targets

- svc-scaffold
- svc-type-check
- svc-package
- svc-upload
- svc-test-remote

Console Output

```
Started by user Jenkins Administrator
Building in workspace C:\Development\Je
[CI Workspace] $ cmd.exe /C "C:\Develop
Buildfile: C:\Development\Jenkins\CI Wo

svc-scaffold:
[svcscaffold] Created new project: C:\D

BUILD SUCCESSFUL
Total time: 0 seconds
[CI Workspace] $ cmd.exe /C "C:\Develop
Buildfile: C:\Development\Jenkins\CI Wo

svc-type-check:
[svctypecheck] Completed Type Check

BUILD SUCCESSFUL
Total time: 2 seconds
[CI Workspace] $ cmd.exe /C "C:\Develop
Buildfile: C:\Development\Jenkins\CI Wo

svc-package:
[svcpackage] Created service archive: C

BUILD SUCCESSFUL
Total time: 0 seconds
[CI Workspace] $ cmd.exe /C "C:\Develop
Buildfile: C:\Development\Jenkins\CI Wo

svc-upload:
[svcupload] Http client: created.
[svcupload] SLF4J: Failed to load class
[svcupload] SLF4J: Defaulting to no-ope
[svcupload] SLF4J: See http://www.slf4j
[svcupload] Request: POST http://192.16
archive/ HTTP/1.1
[svcupload] Response: HTTP/1.1 200 OK
[svcupload] Response: {
```

Scroll down to see the end of the output

5-minute GitLab Tutorial

Here we provide a brief tutorial showing how to use a version control system to support Continuous Integration. We'll show two strategies for storing content in version control, a minimalist source-only method, and a complete build environment method. We'll be basing the tutorial on GitLab.

Prerequisites

You need to have these prerequisites in place before commencing this tutorial:

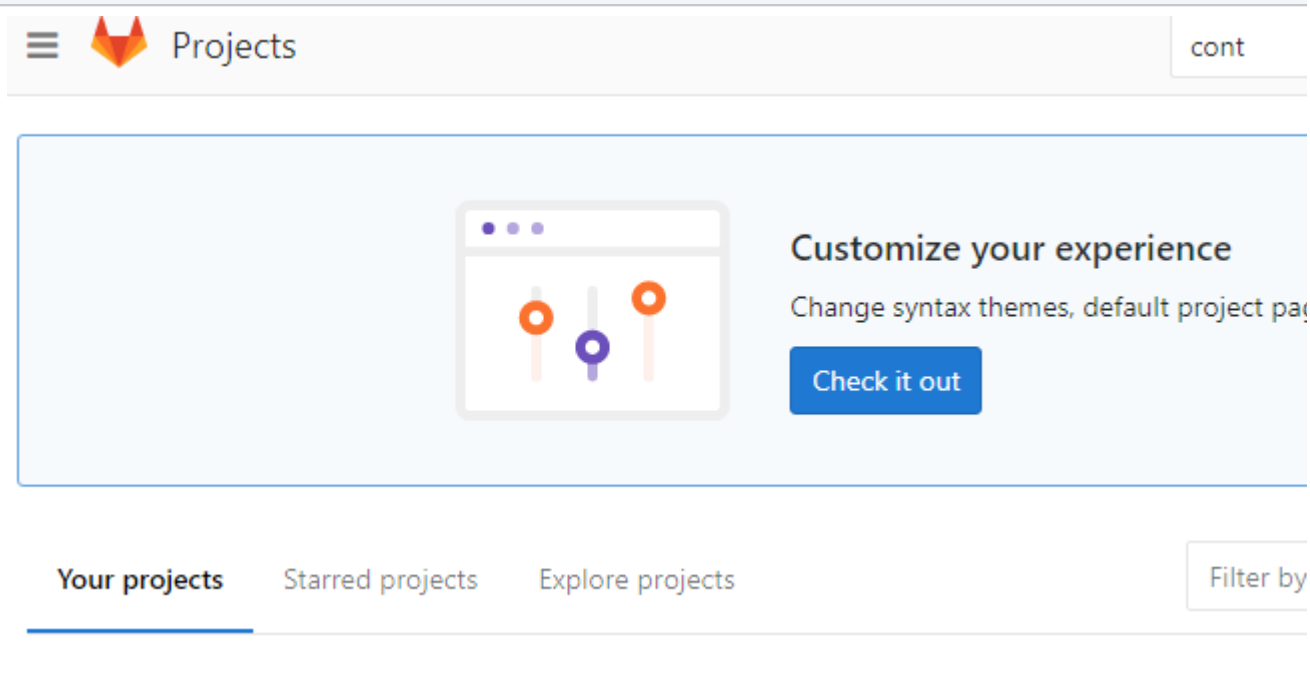
- a login to a GitLab server with enough access rights to create projects, or have an administrator setup a project and .gitignore file for you
- an installed Git client, if you don't already have one, download a Windows client from https://git-scm.com/download/win
- you have already completed the Jenkins Tutorial

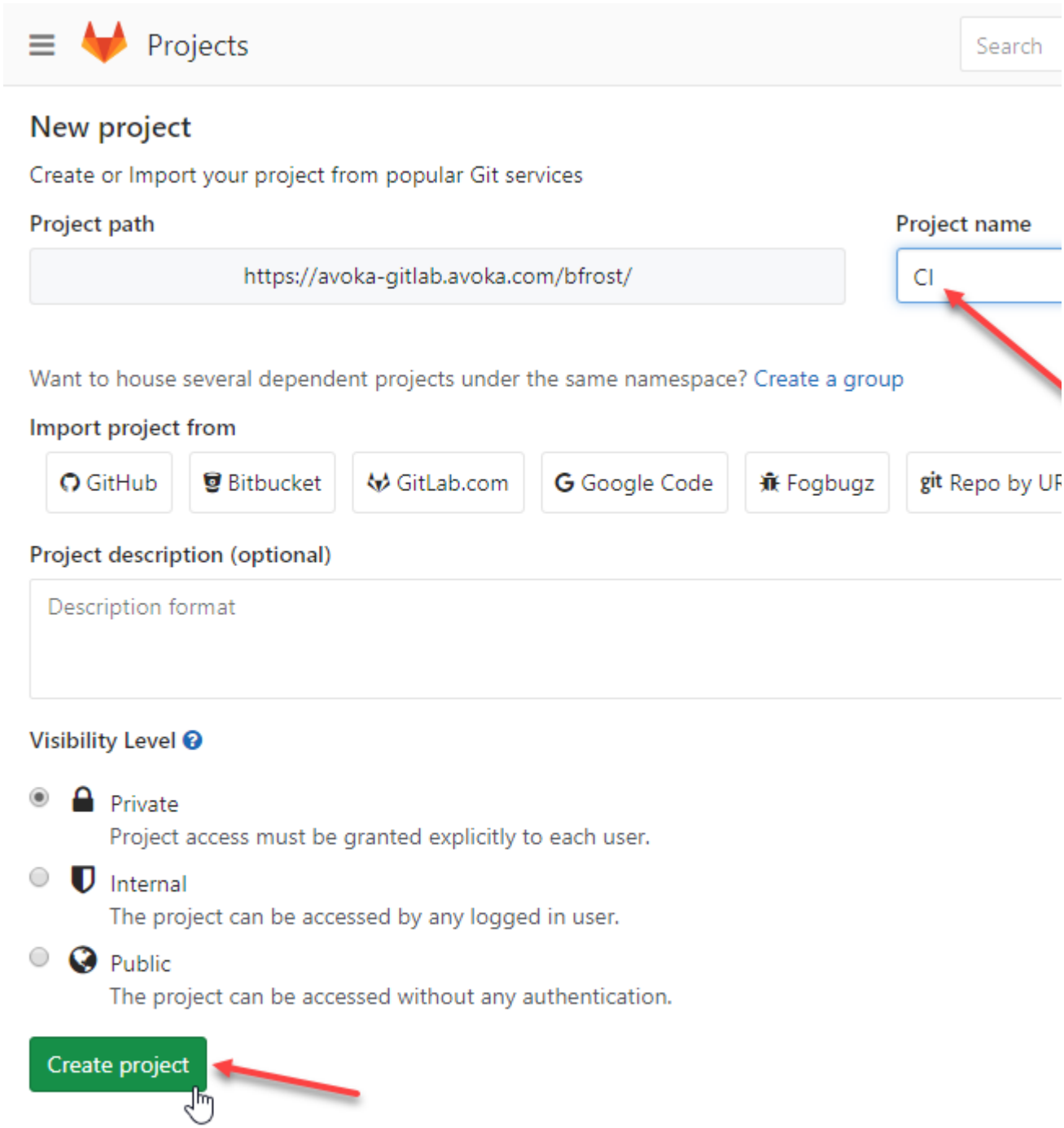
Recipe

In this tutorial, we will:

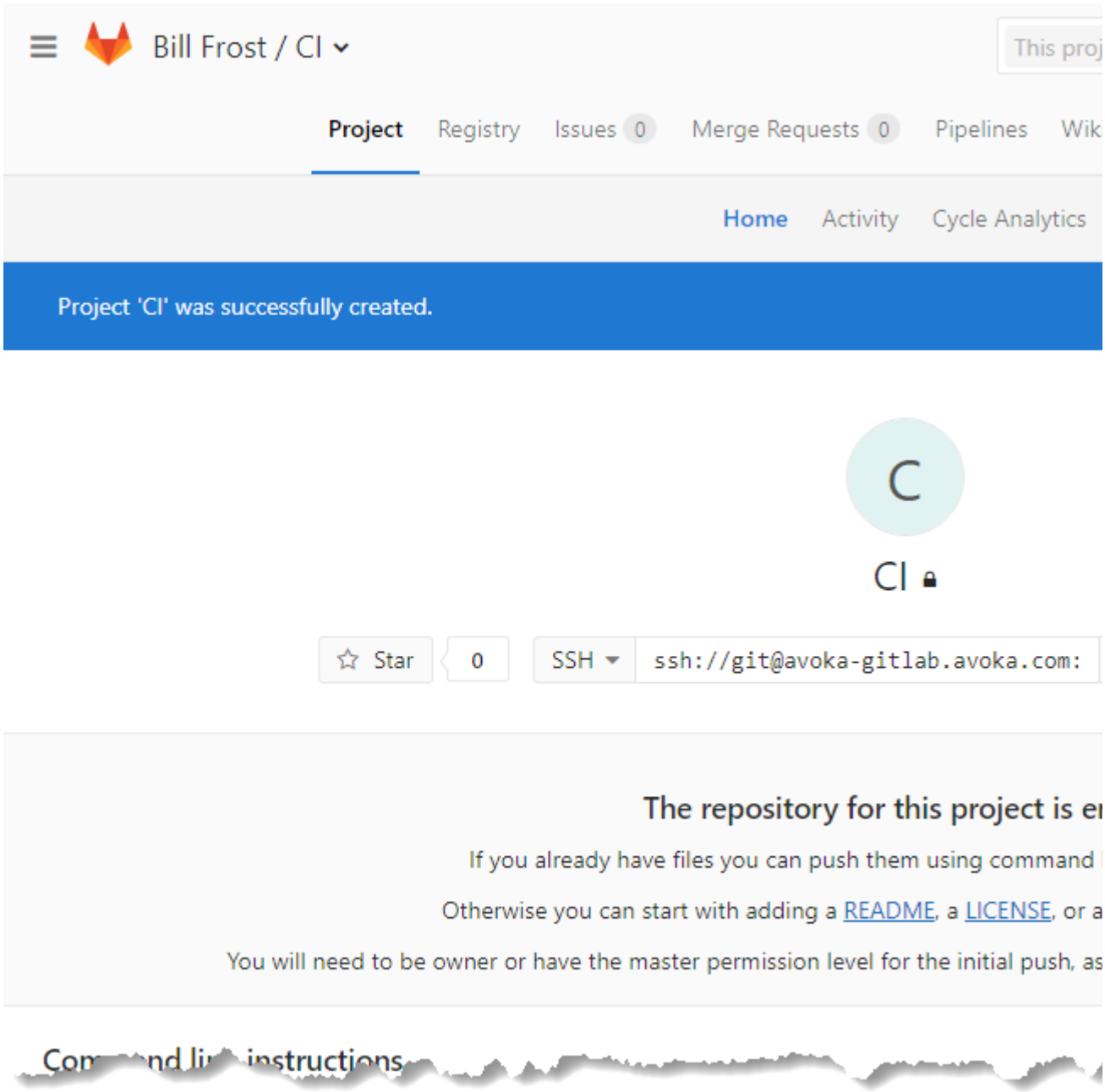
1. Create a GitLab project on a GitLab server
2. Create a .gitignore file in the project to ignore all but the src/* folder in our project, providing the minimalist approach
3. Initialize a local GitLab repository in an existing Jenkins CI workspace containing the Transact Fluent SDK
4. Clone the GitLab project to the local repository
5. Check-in the local artifacts generated by the Transact Fluent SDK
6. Hook the Jenkins CI project to the GitLab project
7. Run the Jenkins CI project to pull from GitLab and execute the CI pipeline
8. Alter the .gitignore file to add all files to GibLab, except build.properties, providing the complete approach
9. Push all files to GitLab
10. Create a build.properties-TEMPLATE file so the GitLab project contains a template for new build.properties files
11. Push the new template file to GitLab

Step-by-step Instructions - Minimal Version Control

Step	Screenshot/Details
Login to GitLab Click New Project	
Enter a name and click Create Project	

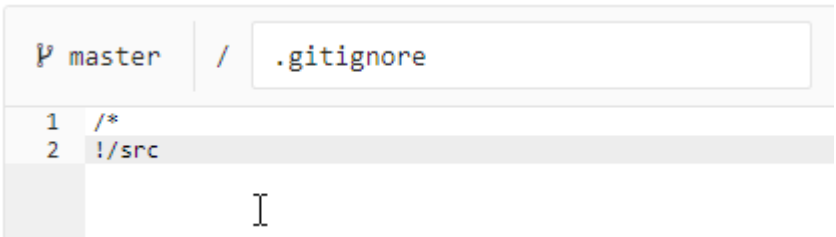


Click on the .gitignore link



Enter or copy /paste these values

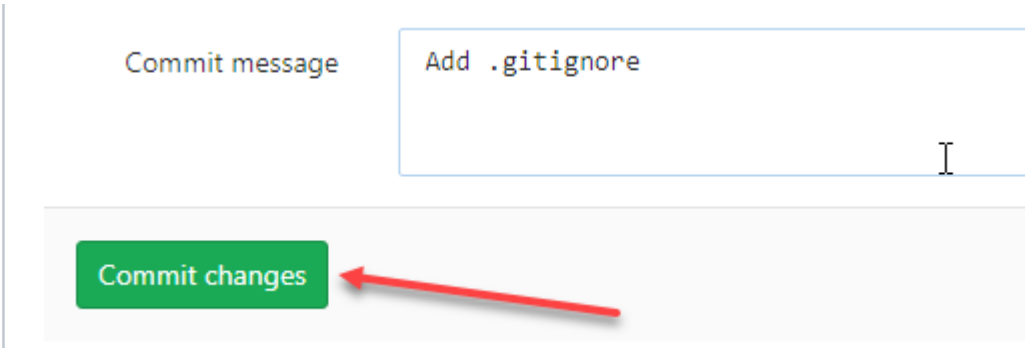
These values will cause commits to ignore all files except those in the src folder



Copy/Paste .gitignore text

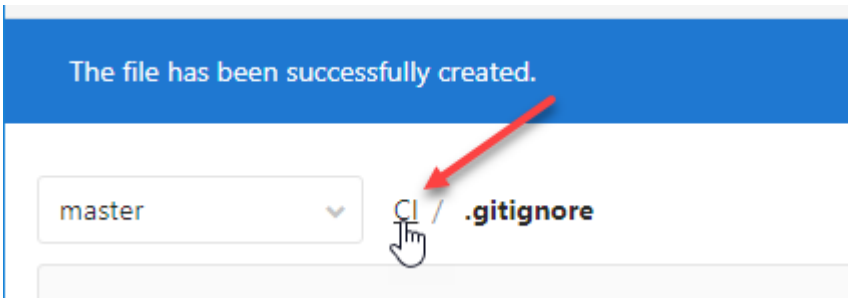
```
/*
!/src
```

Click Commit changes

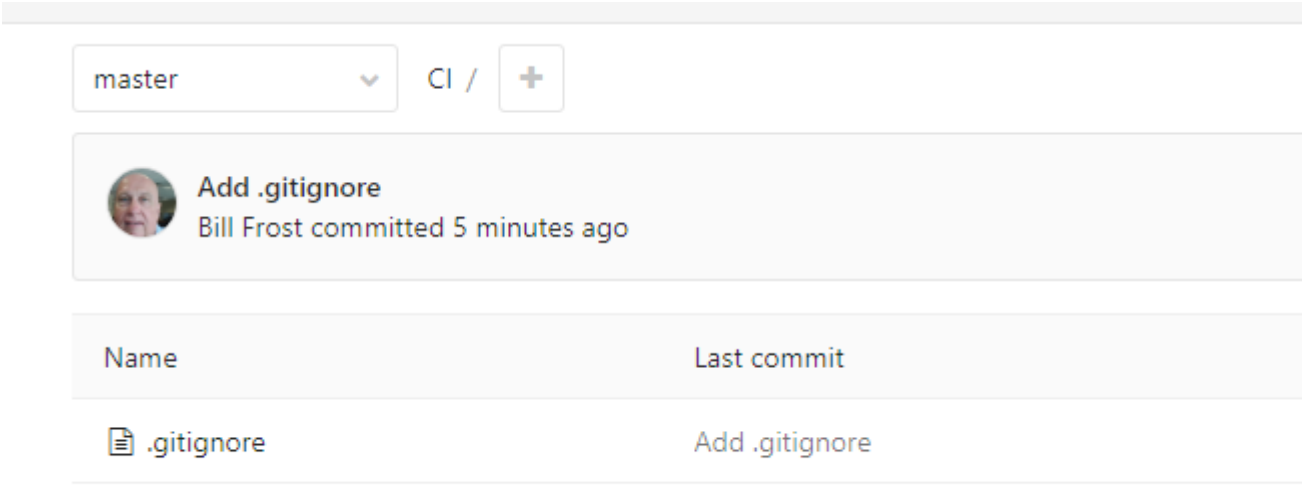


The file will be created

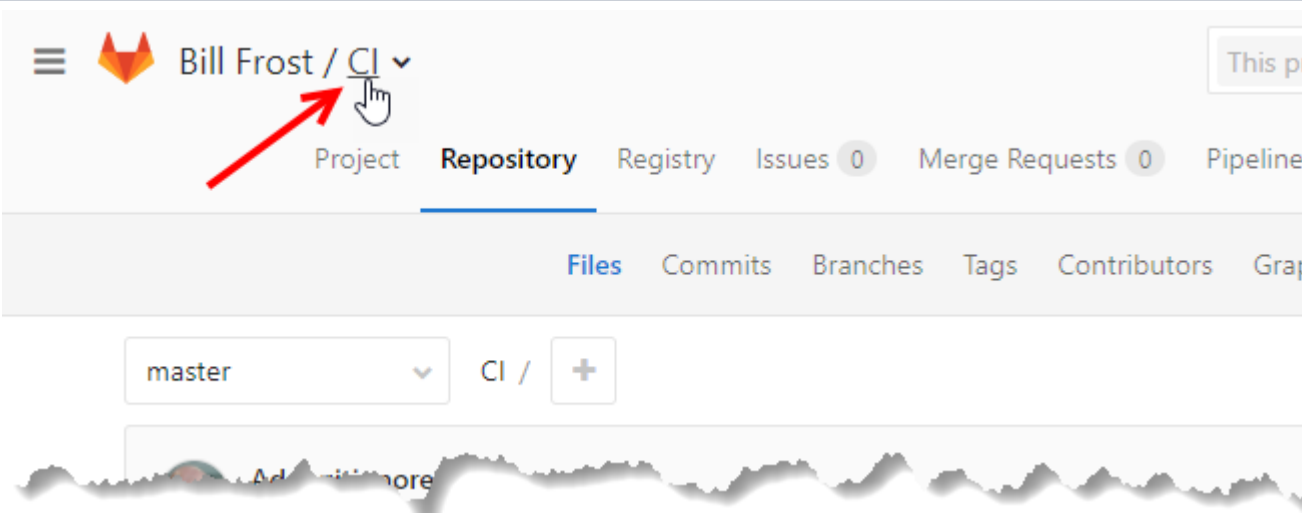
Click the project name part of the breadcrumb



You will see the file has been committed

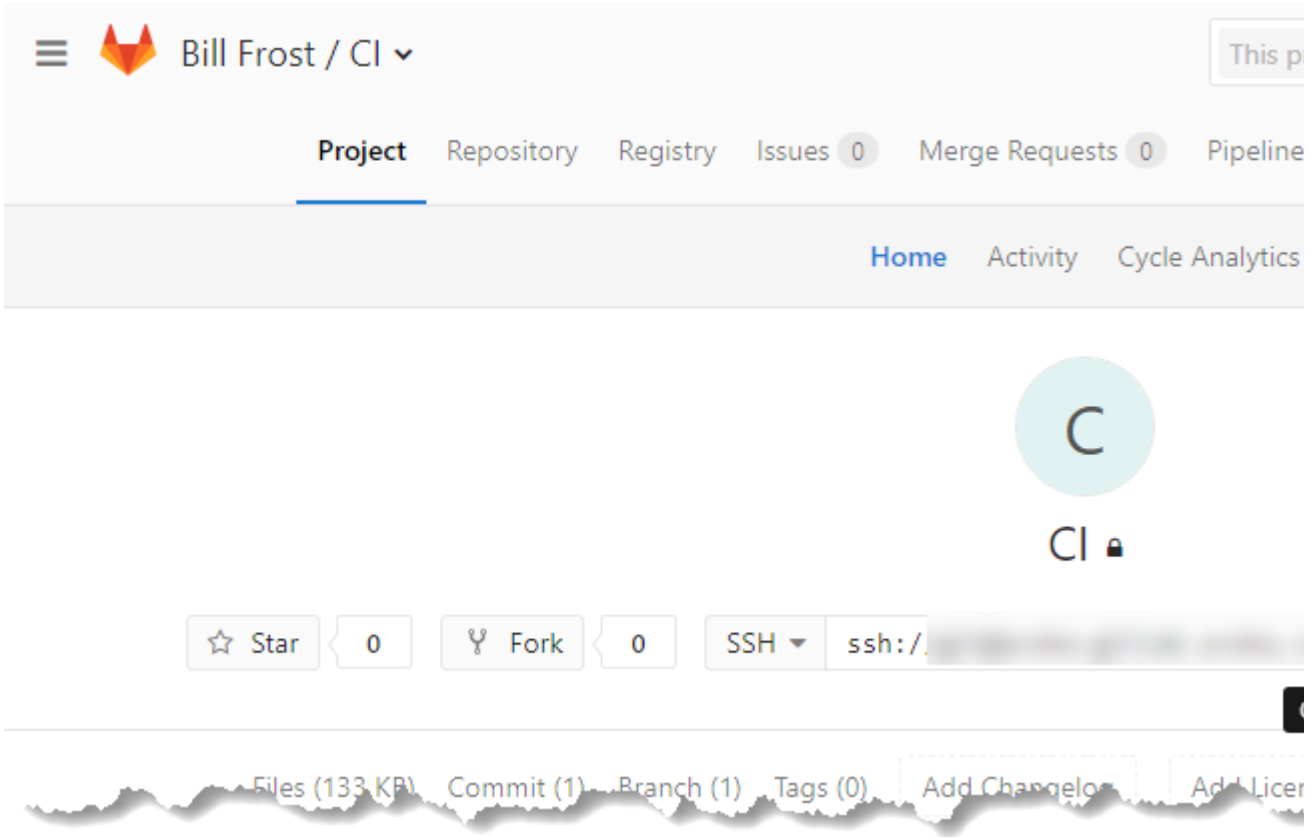


Click on the project name breadcrumb in the header



Copy the project ssh URL to the clipboard to use in the

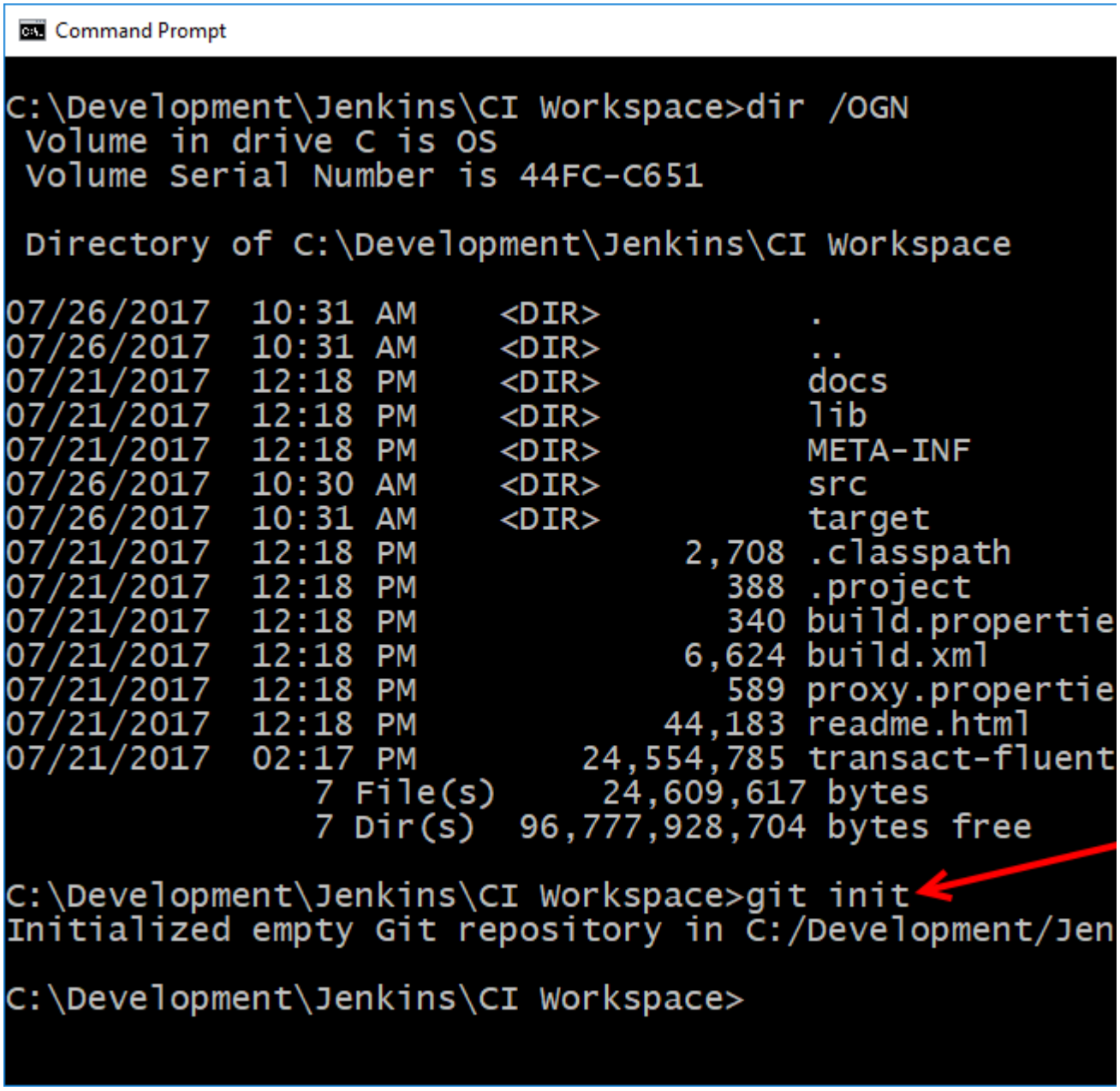
following commands



Open a CMD prompt and cd to the CI Workspace folder created in the previous tutorial

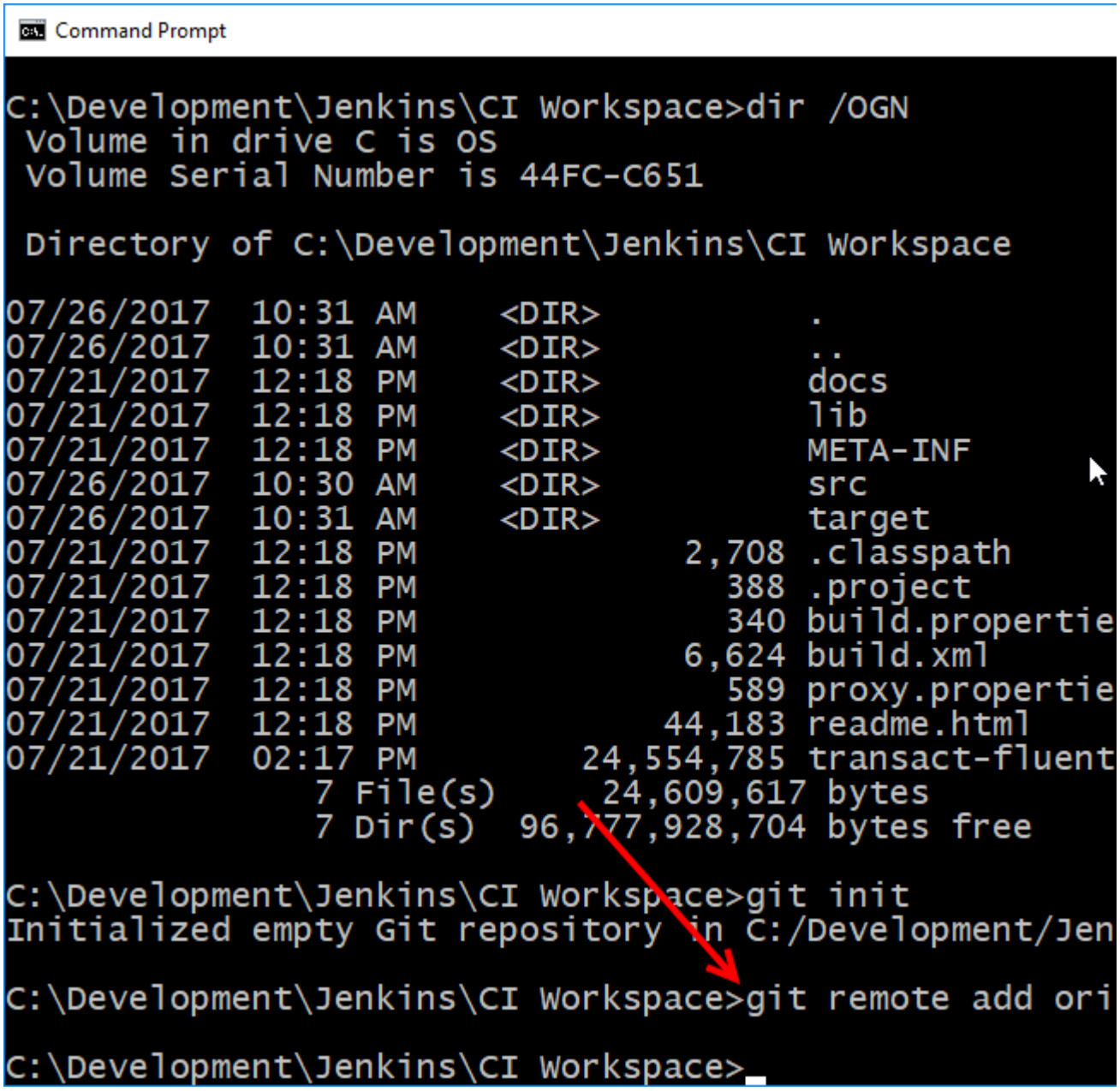
Type 'git init' and enter

This will establish the current directory as a local Git repository



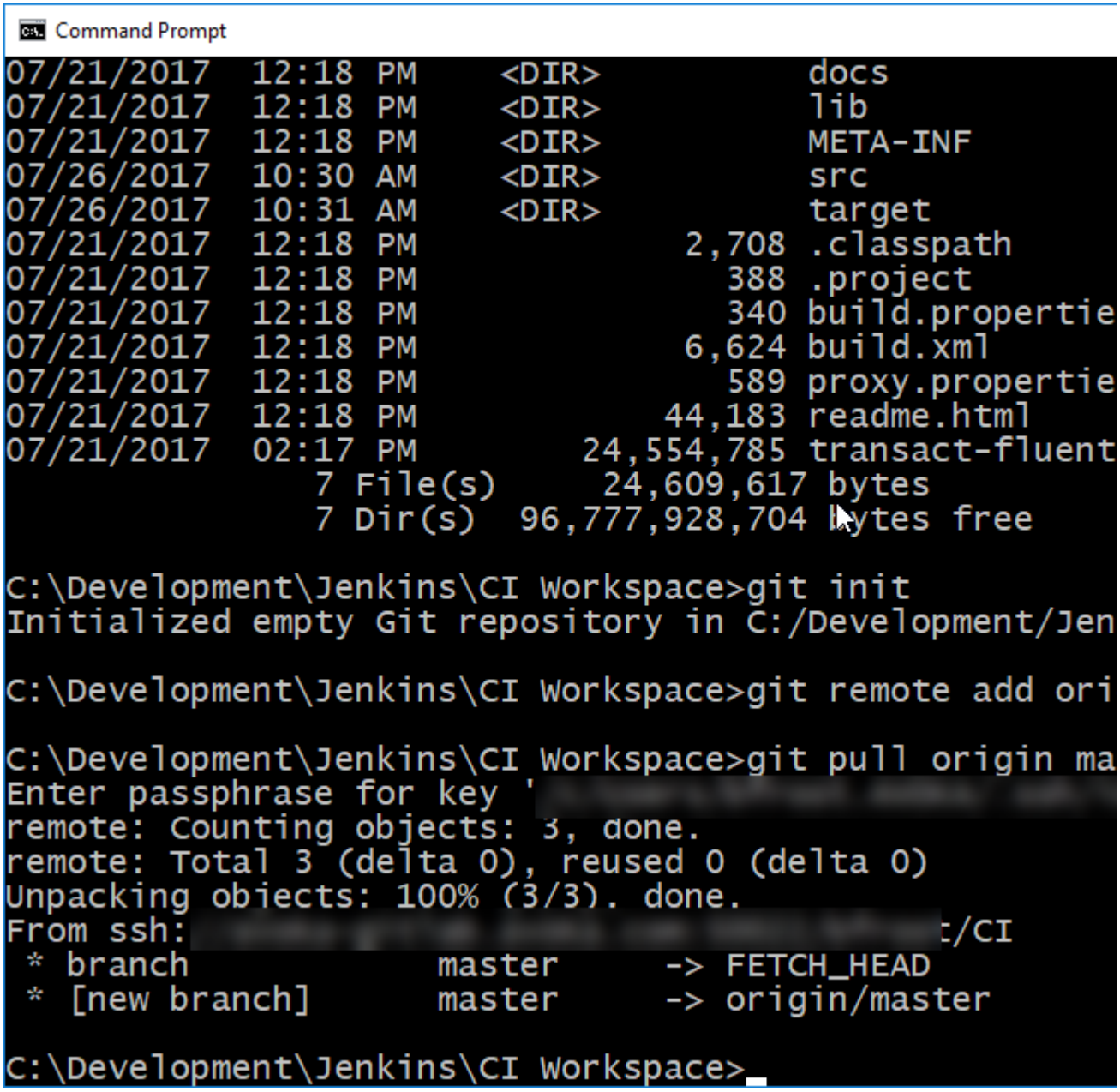

```
git init
```

Enter 'git remote add origin ' and your Git project URL, and enter



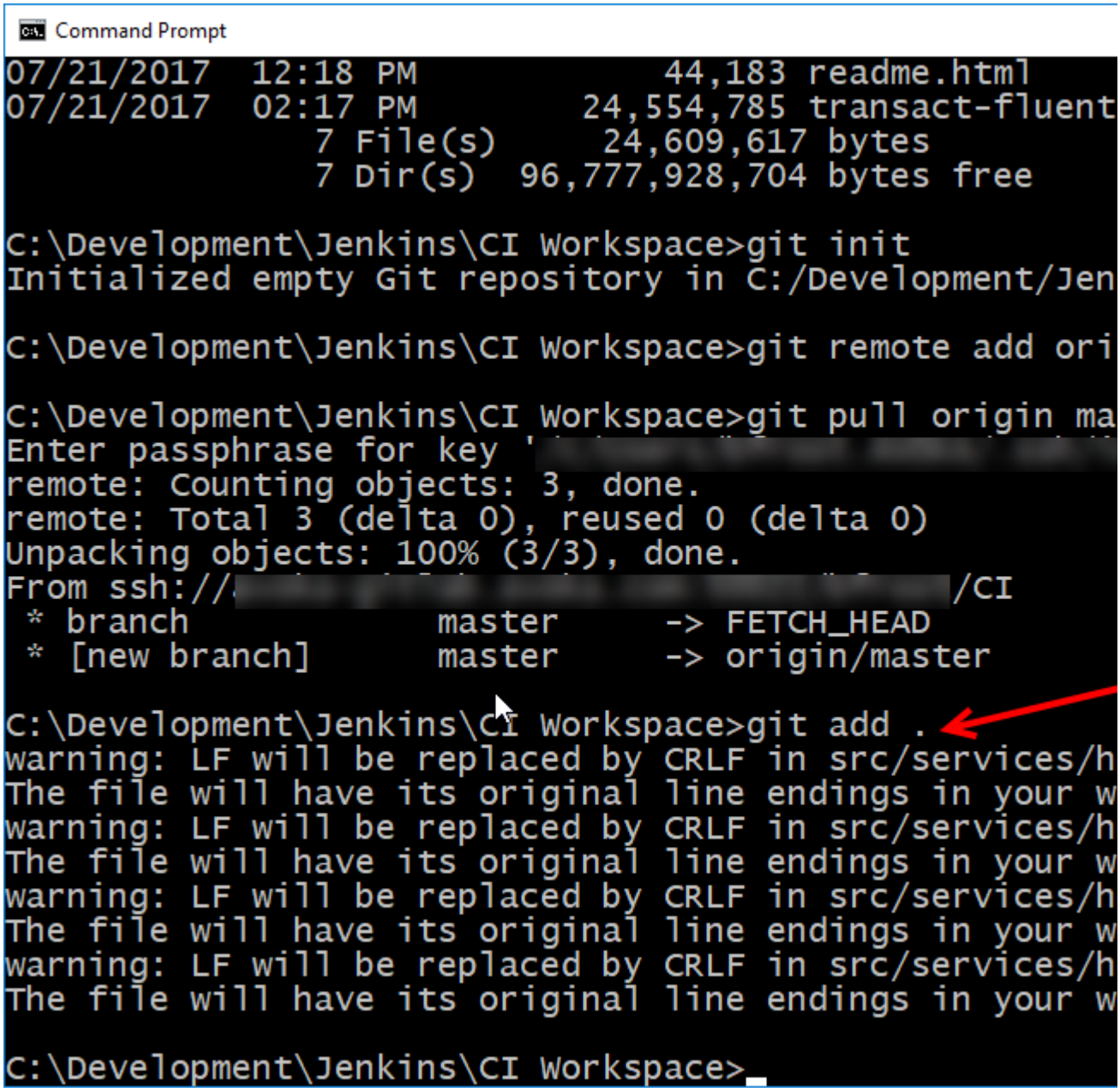

```
git remote add origin <your Git project URL>
```

Type 'git pull origin master ' and enter



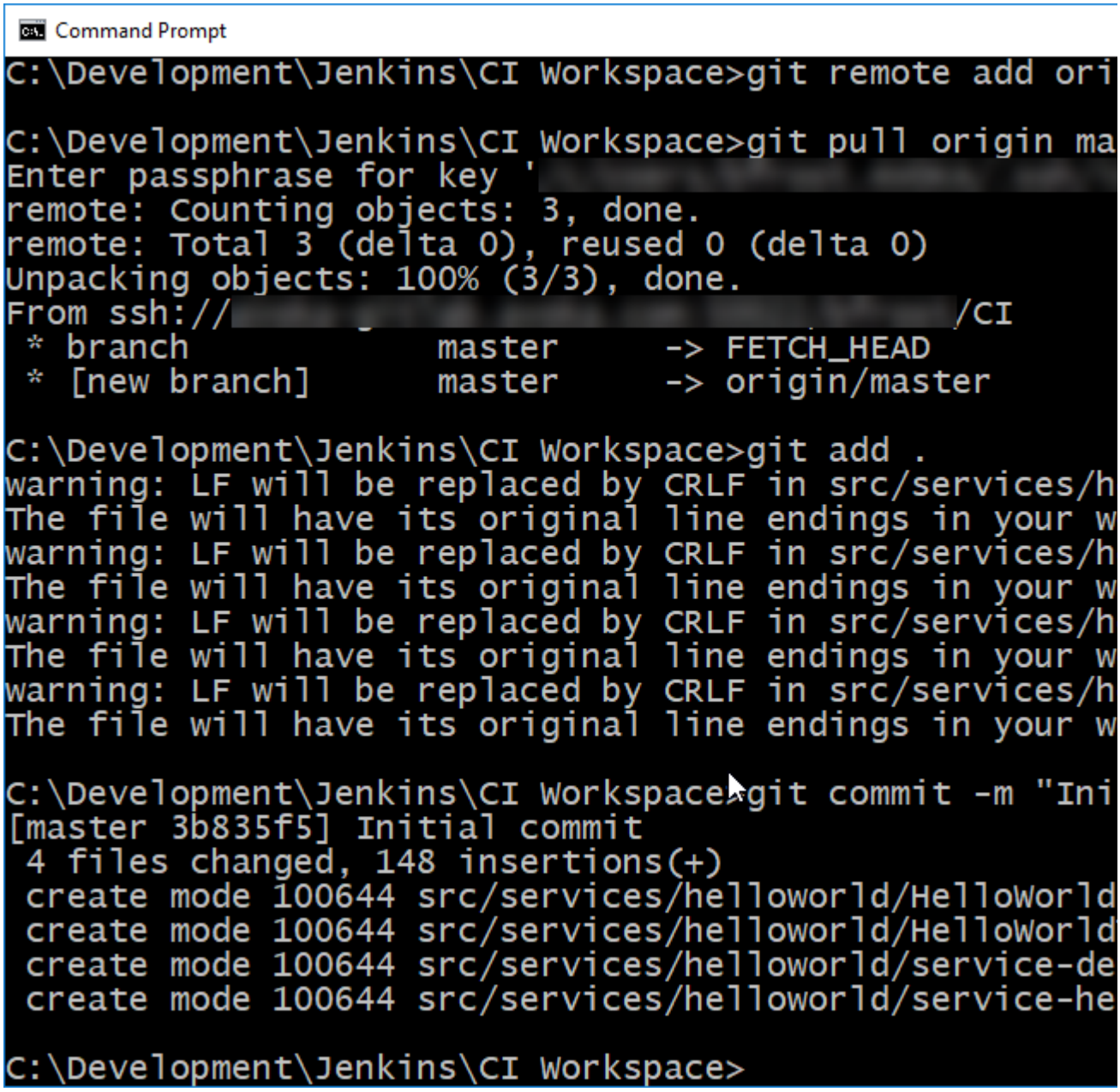

```
git pull origin master
```

Type 'git add .' and enter




```
git add .
```

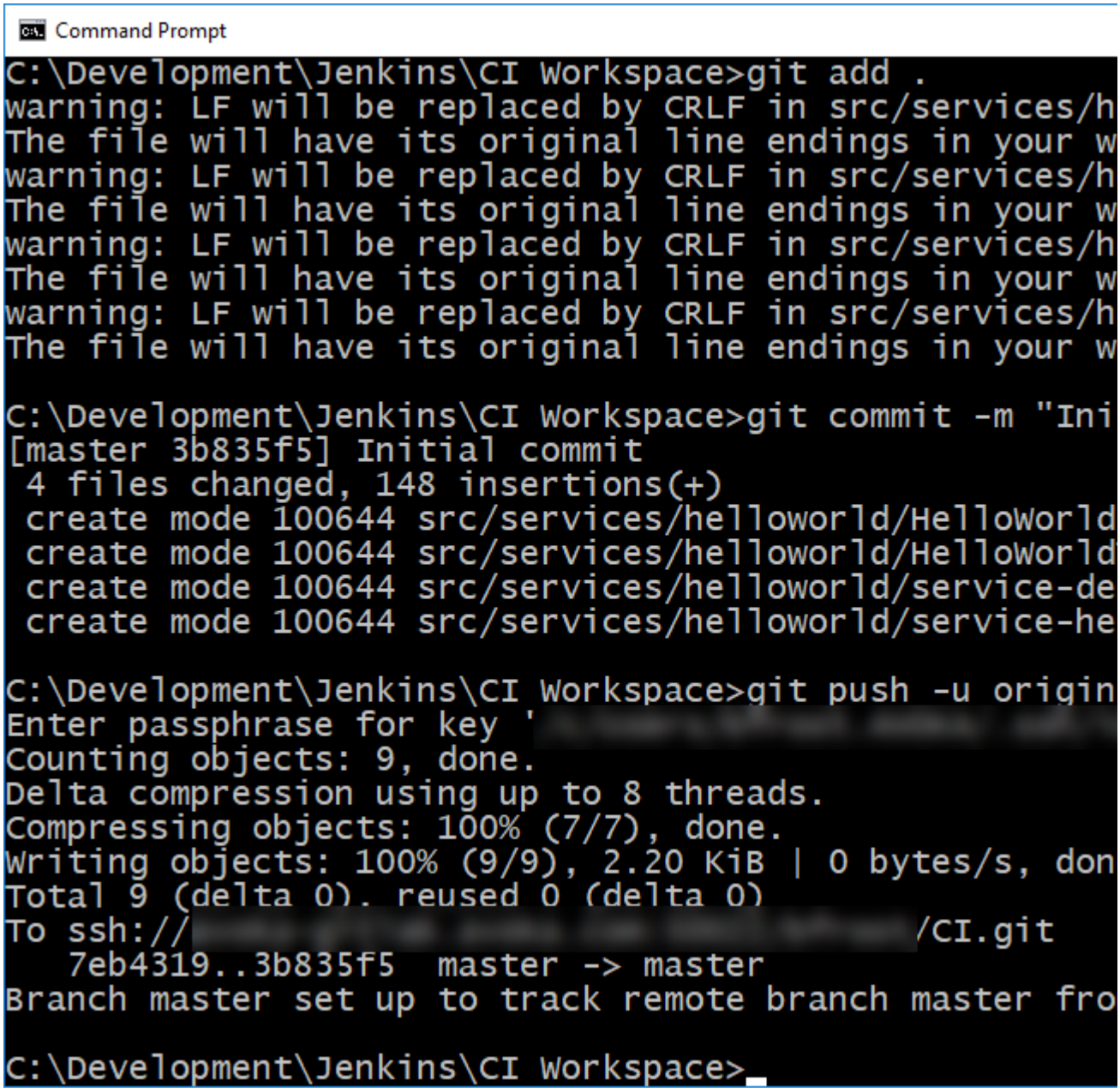
Type 'git commit -m "Initial commit"' and enter




```
git commit -m "Initial commit"
```

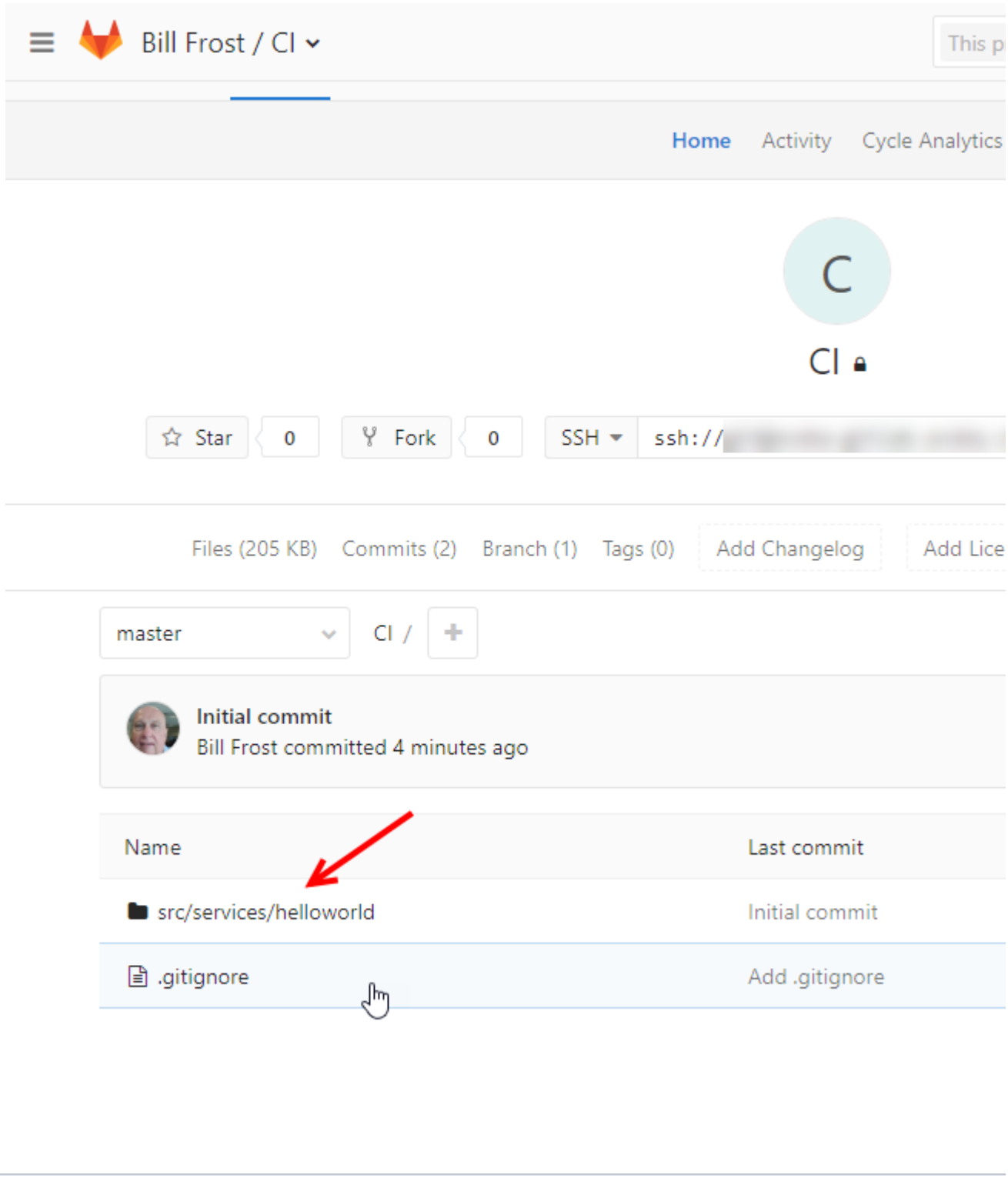
Type 'git push -u origin master ' and enter

The local files that match the .gitignore rules have now been stored in the GitLab project




```
git push -u origin master
```

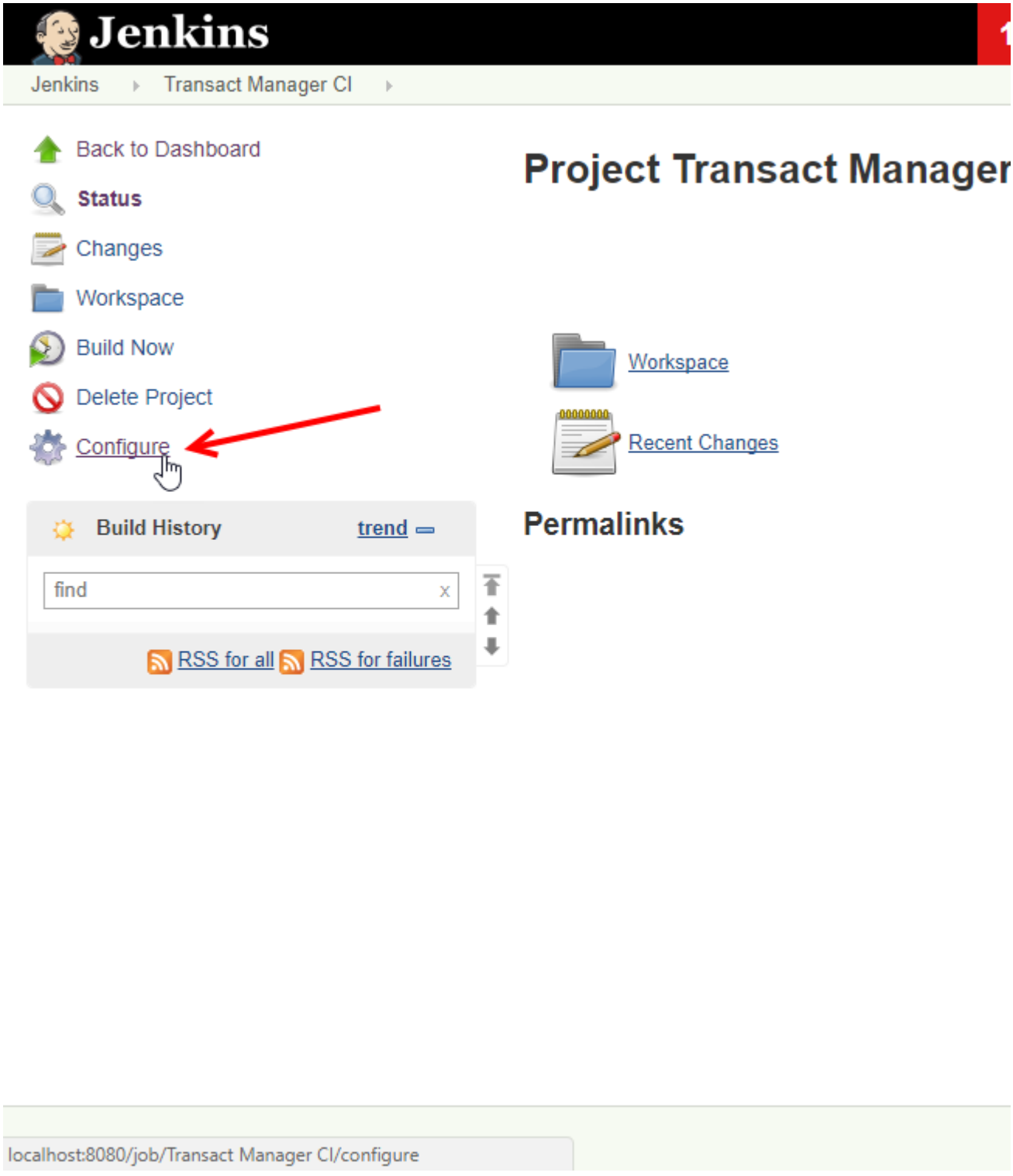
Refresh the GItLab project page and you will see the src folder and its children are now stored in the project



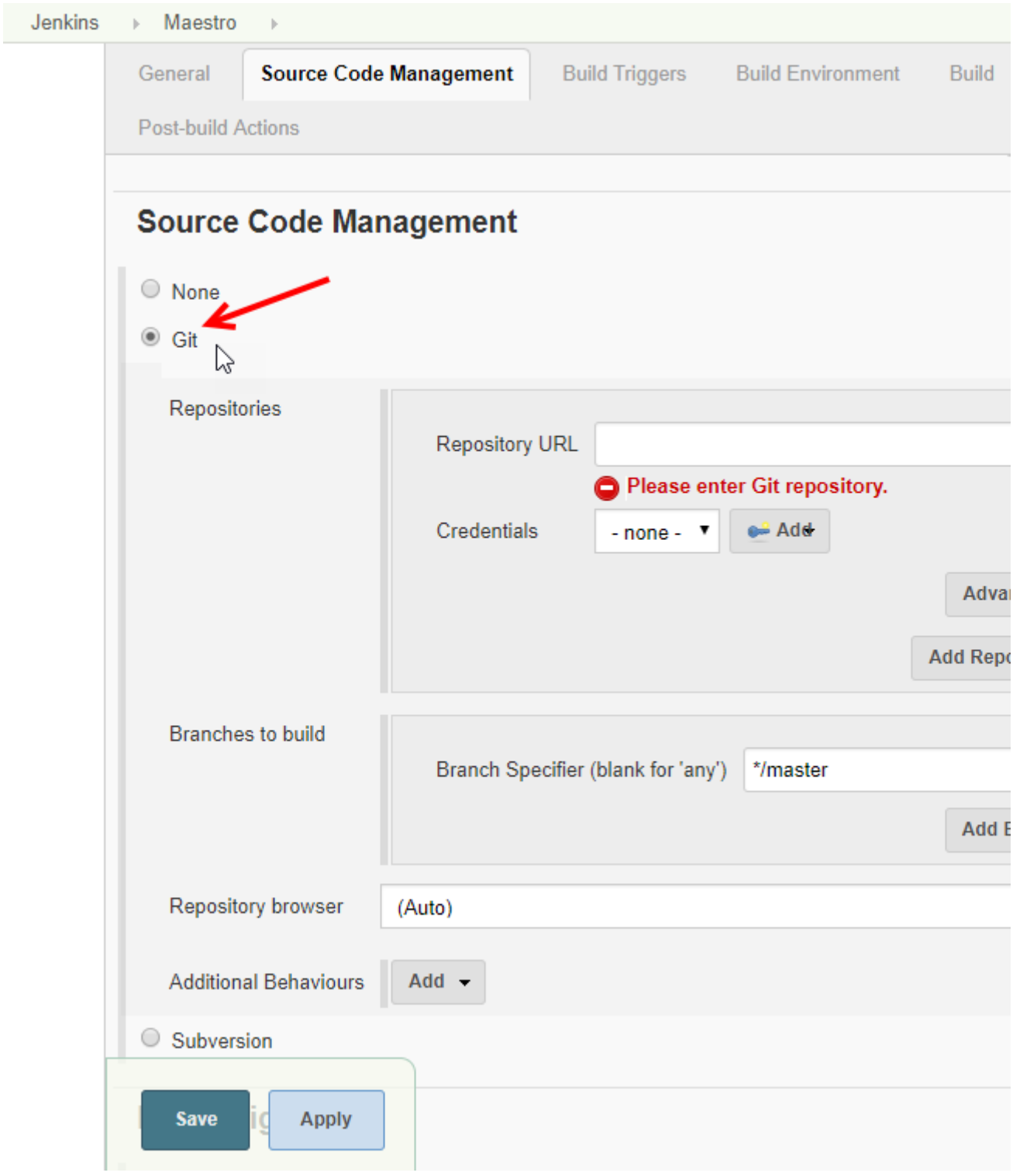
Now we'll configure Jenkins to poll GitLab for changes, and trigger a build if any are found

Return to the Jenkin

s project and click the Configure link



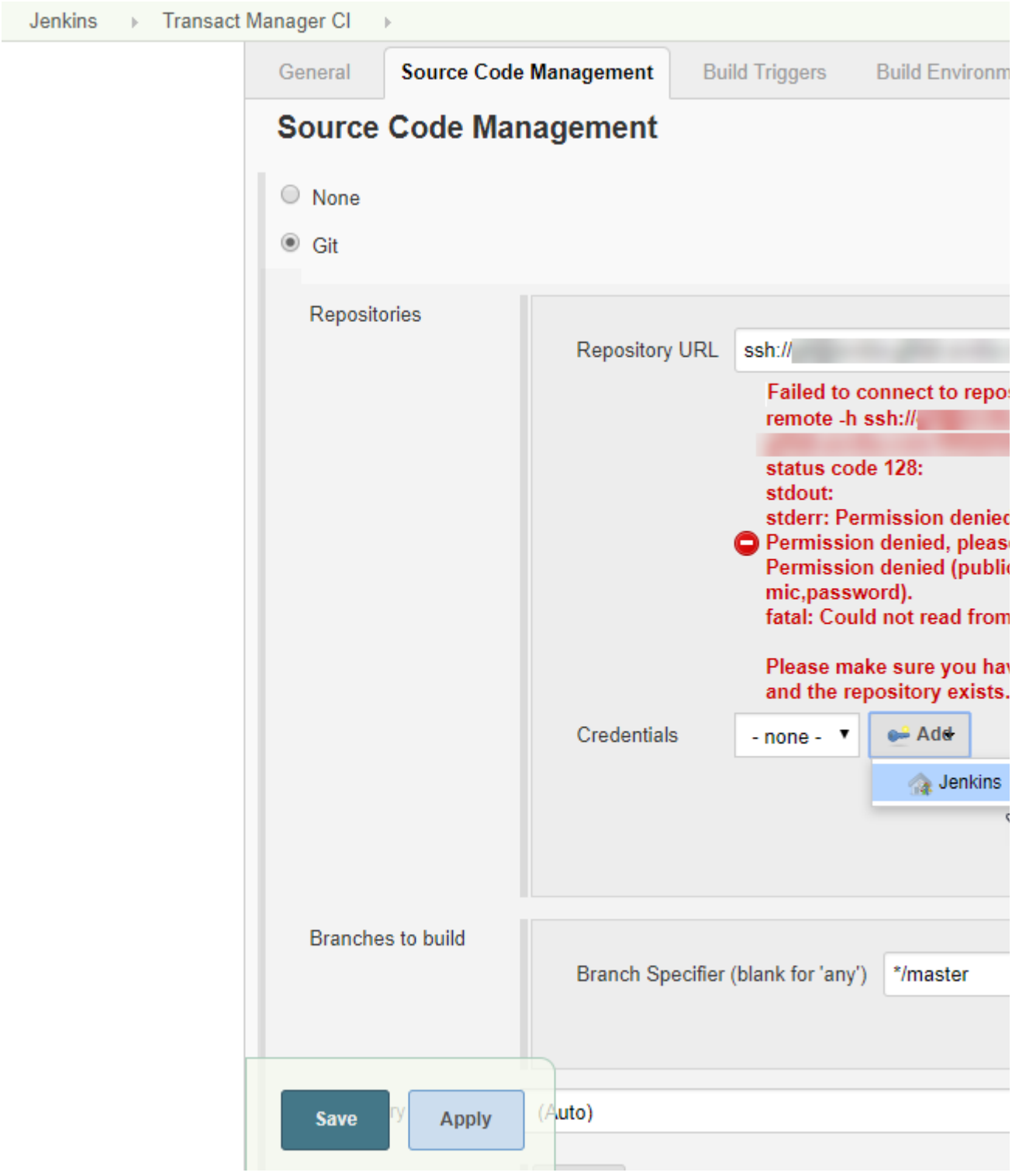
Scroll down to Source Code Management and click the Git radio button



Enter the GitLab project URL and tab out of the field - red credential error messages will appear

Click Add and

select Jenkins to add a new credential

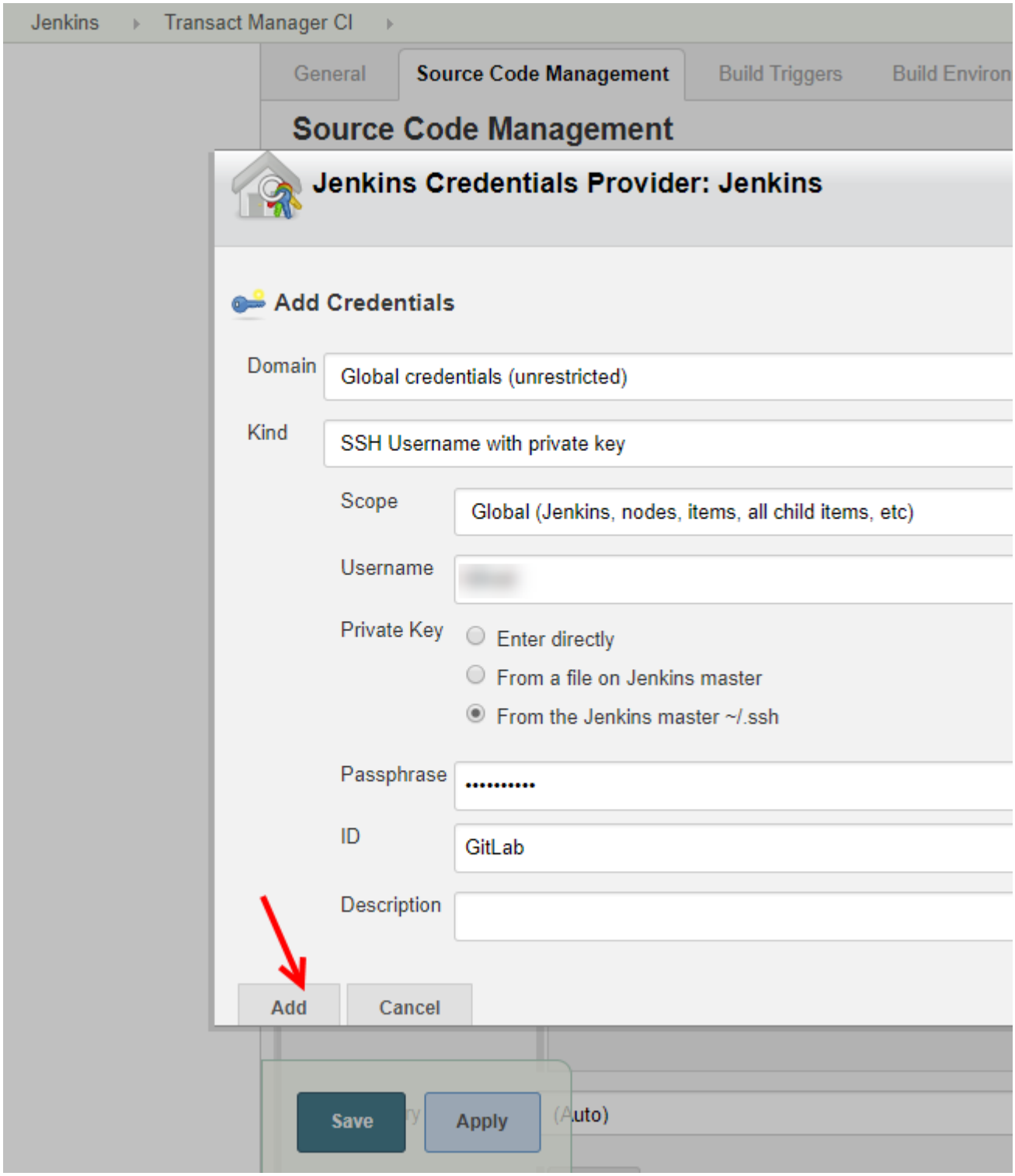


Select the Kind of credential to add

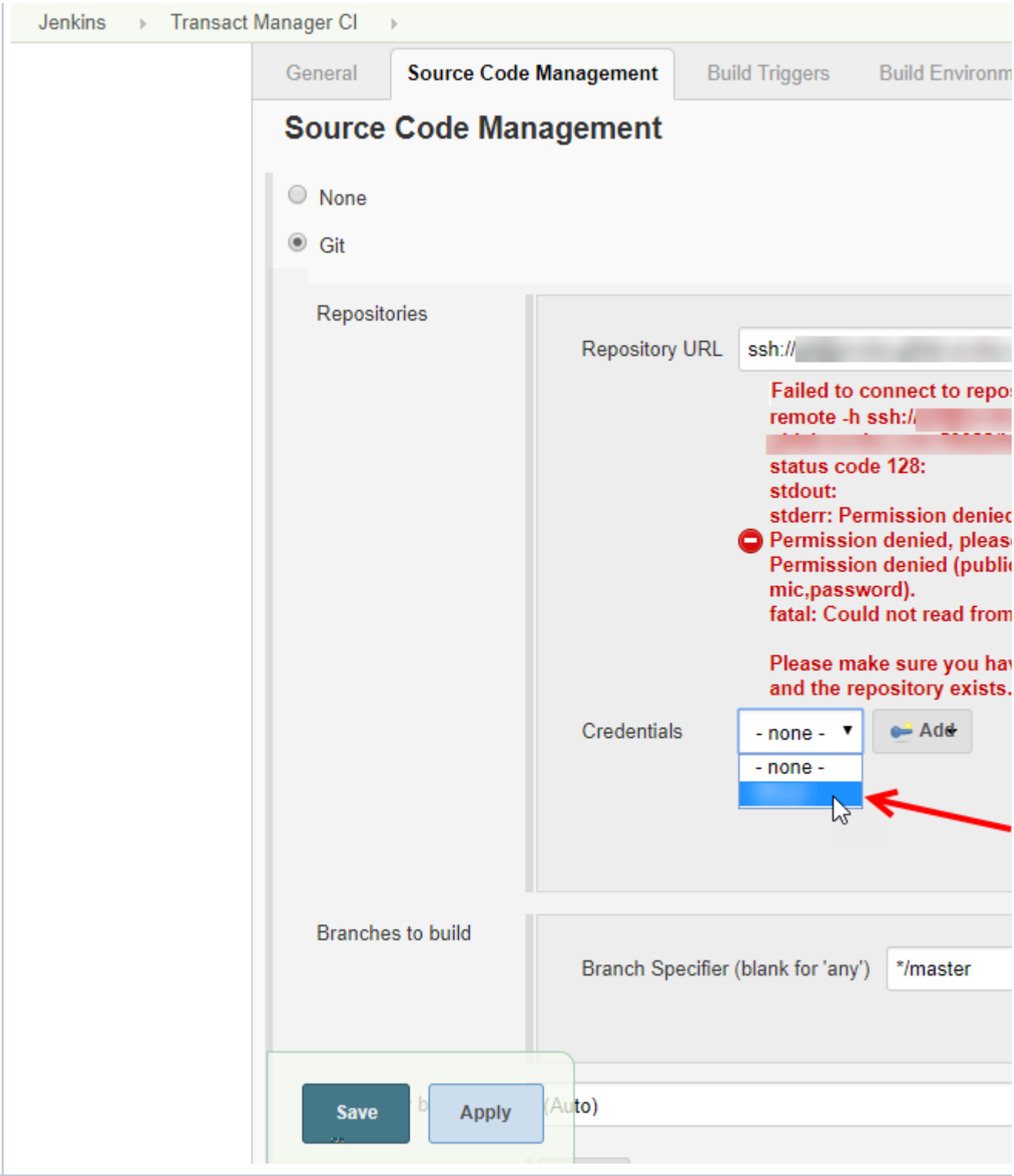
Enter additional fields according to the credential Kind chosen - the

dialog will vary

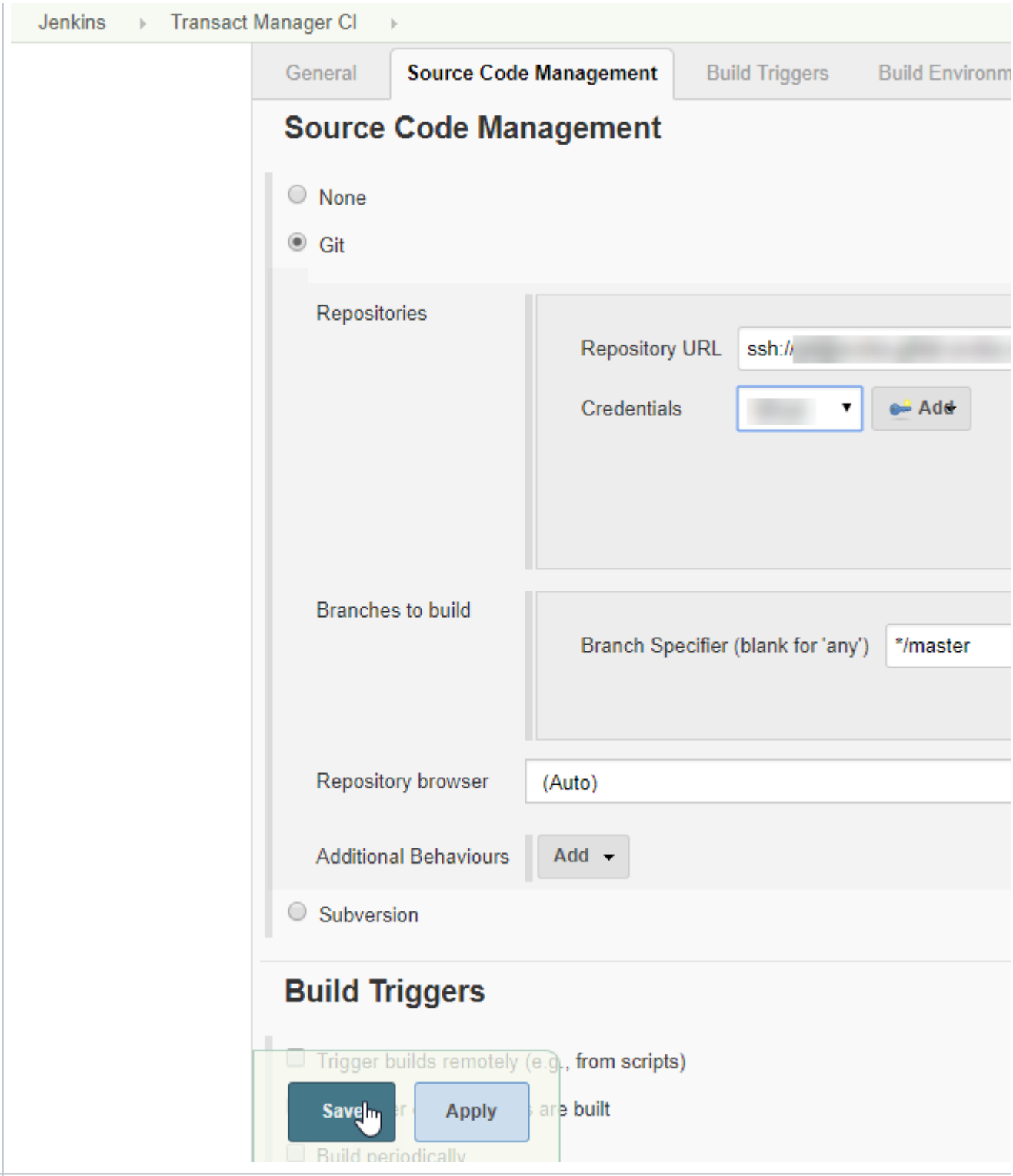
Click Add



Back on the Source Code Management block, select the newly added credential from the dropdown



The GitLab credentials should validate and the red errors disappear



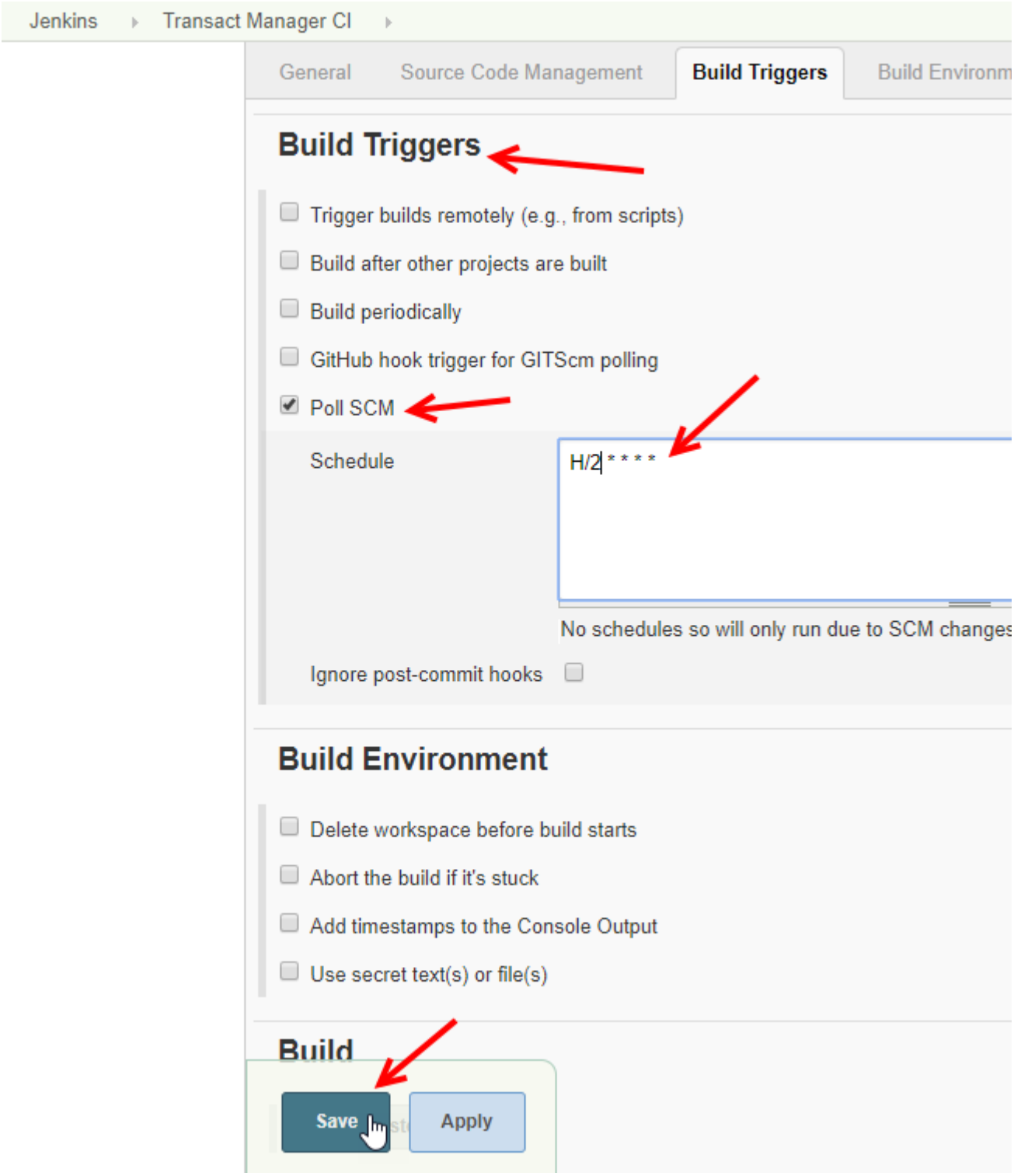
Scroll down to Build Triggers

Click Poll SCM

Enter 'H/2 * * * *' in the Schedule text area: this will poll GitLab

every 2 minutes

Click Save and you will be returned to the project dashboard

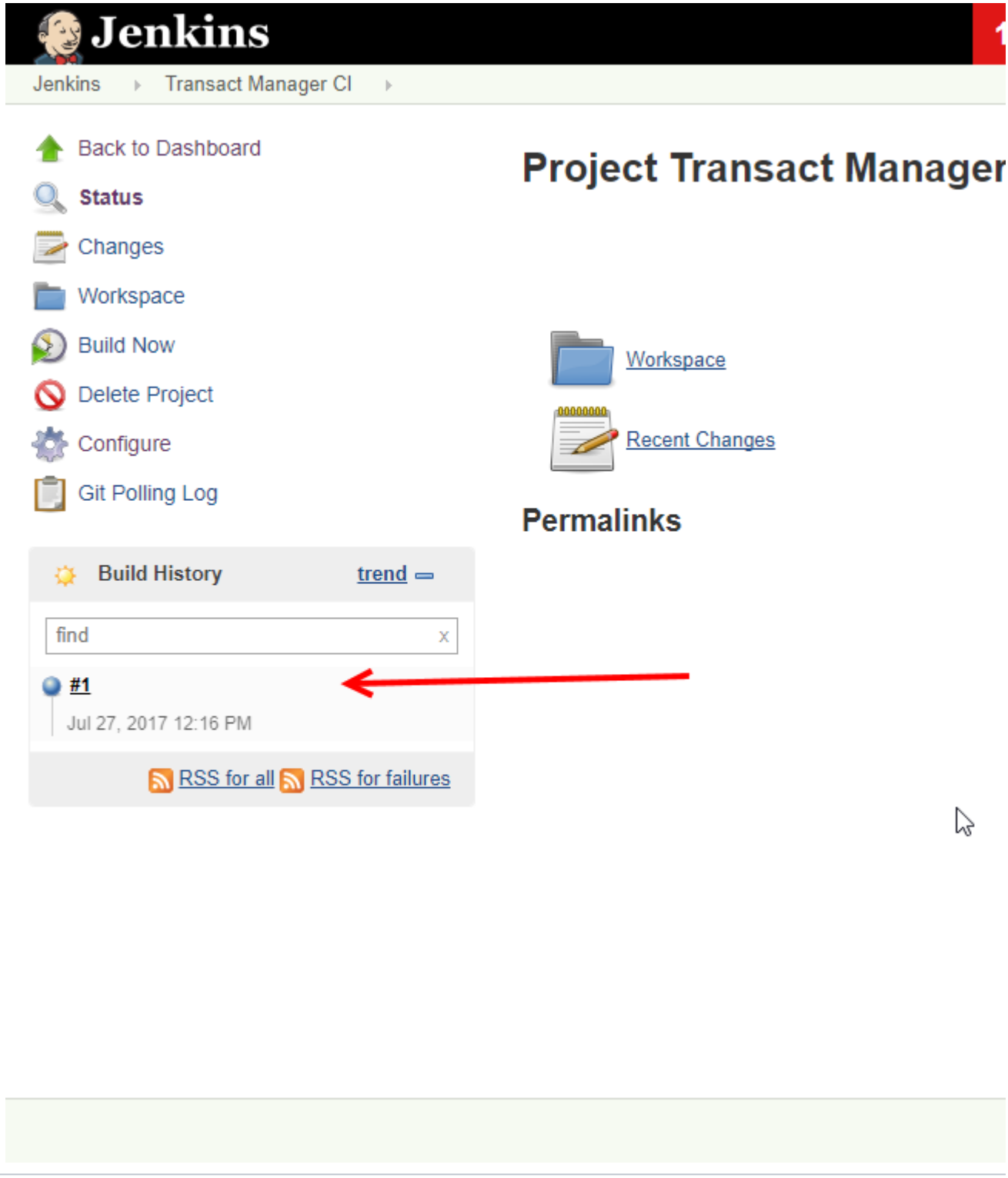


Now you can change a local file and check it into GitLab

Enter these commands from a CMD prompt

Git commands to check in changes

```
cd "C:\Development\Jenkins\CI Workspace"
edit src\services\helloworld\service-help.html
  (change something and save the file)
git add .
git commit -m "Triggering Jenkins Build"
git push -u origin master
```

Within 2 minutes, a new build will be triggered and you will see that shown on the Build History panel of the project dashboard	

Additional Instructions - Complete Version Control

At this point, you will have a GitLab project populated only with the files you have created locally. It will not contain the Transact Fluent SDK files, because we ignored them with the rules we created in .gitignore. This minimalist method does save space in the GitLab repository, but it may not be ideal in terms of deploying to a new server. Any new server would need to have the Transact Fluent SDK installed on it before pulling the GItLab project.

We can convert the minimalist approach to a complete approach simply by changing the .gitignore file, adding additional files and committing to the GitLab project. Now when the project is pulled from GitLab, the complete Transact Fluent SDK comes with it, so it can be deployed and is immediately ready to go on a new server.

We want to add all files from our local folders to GitLab, but we want to exclude the build.properties file, because it contains sensitive passwords.

Steps

Step	Screenshot/Details
Edit	

the contents of your local .gitignore file

.gitignore Contents

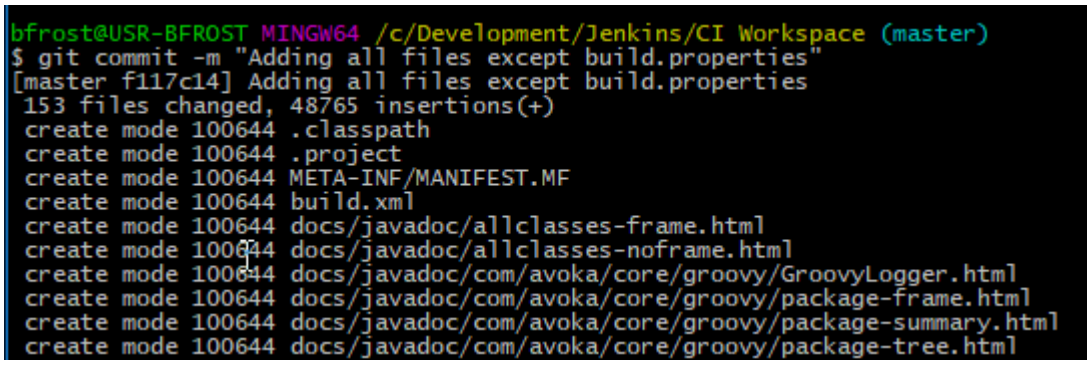
```
/build.properties
```

Add new files to GitLab

.gitignore

```
git add .
git commit -m "Adding all files except build.properties"
git push -u origin master
```

You will see a long list of filenames added by the git commit command



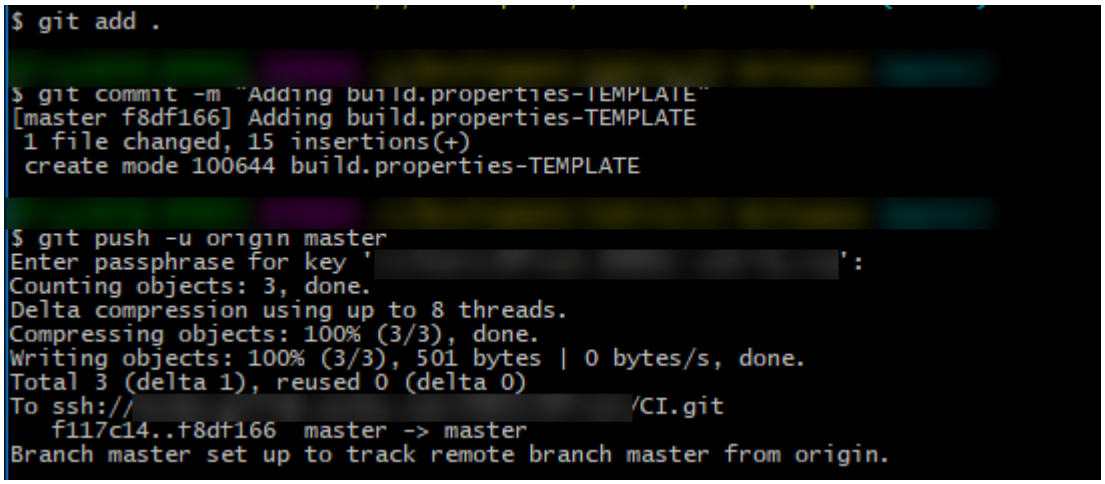
Now we have all files in the GitLab project, except build.properties

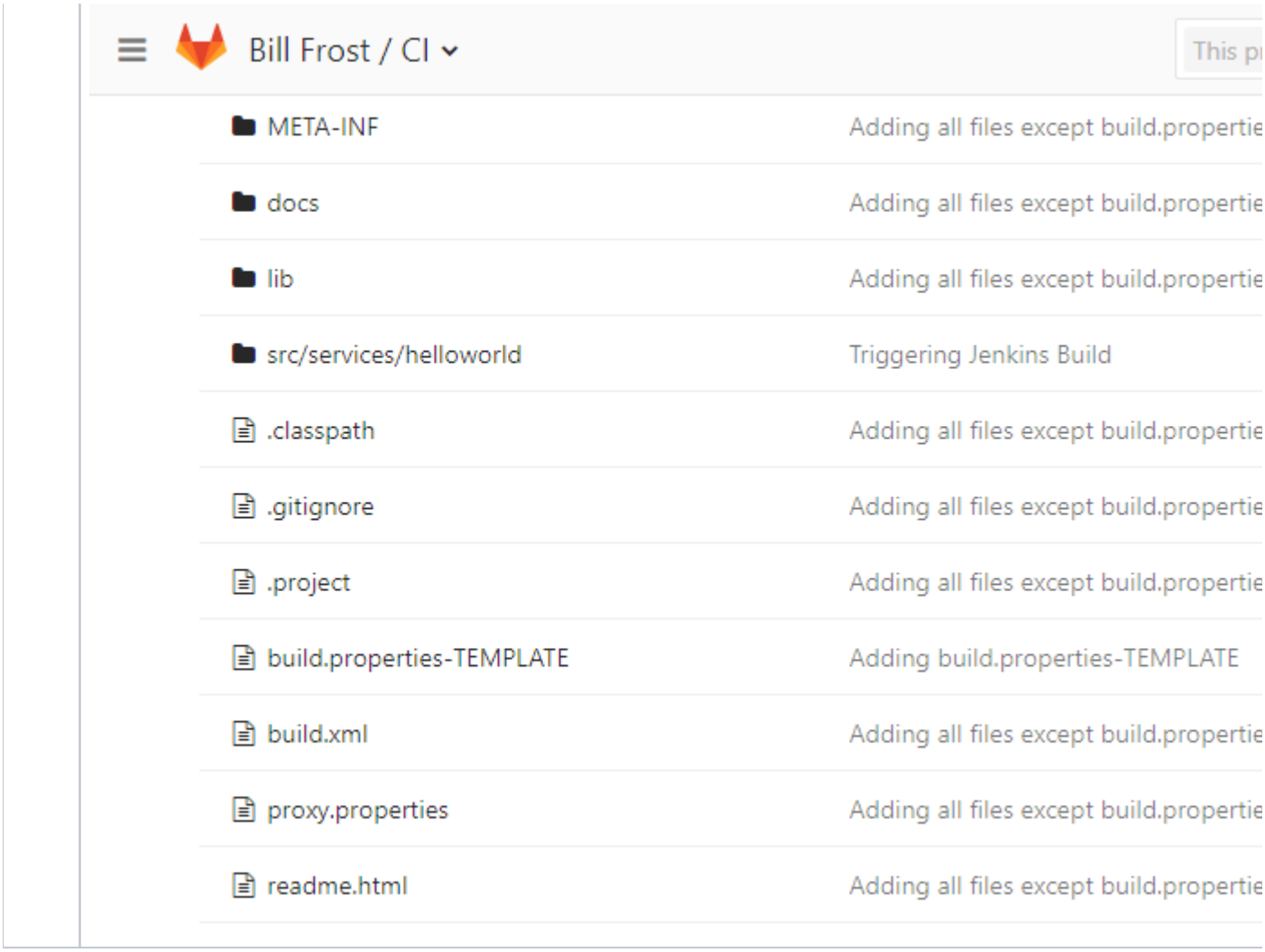
We now also don't have any idea what goes in the build.properties file so we should add a build.properties-TEMPLATE file that contains the default build.properties file from the SDK

Extract build.properties from the Zipped

.gitignore

```
git add .
git commit -m "Adding build.properties-TEMPLATE"
git push -u origin master
```

Transact Fluent SDK file, and rename it to build.properties-TEMPLATE	
Now our GitLab project contains all of the Transact Fluent SDK files, except for our customized build.properties with the sensitive passwords, but it does have a build.properties-TEMPLATE file so a new user can use it to construct their own build.properties file.	
Your GitLab project should now look like this	


Bill Frost / CI
This p
META-INF
Adding all files except build.propertie
docs
Adding all files except build.propertie
lib
Adding all files except build.propertie
src/services/helloworld
Triggering Jenkins Build
.classpath
Adding all files except build.propertie
.gitignore
Adding all files except build.propertie
.project
Adding all files except build.propertie
build.properties-TEMPLATE
Adding build.properties-TEMPLATE
build.xml
Adding all files except build.propertie
proxy.properties
Adding all files except build.propertie
readme.html
Adding all files except build.propertie

Transact Fluent API Reference

com.avoka.core.groovy
- GroovyLogger

com.avoka.tm.http
- DeleteRequest
- GetRequest
- HttpRequest
- HttpRequest.FileParam
- HttpRequest.Param
- HttpResponse
- PatchRequest
- PostRequest
- PutRequest
- RequestBuilder

com.avoka.tm.job
- ActionResult
- ActionResultBuilder
- Jobs

com.avoka.tm.query
- JobQuery
- PropertyQuery
- SpaceQuery
- SvcConnQuery
- SvcDefQuery
- TxnQuery
- UserQuery

com.avoka.tm.security
- Saml2Parser
- Saml2ParserResult
- SsoAuthToken

com.avoka.tm.svc
- DeliveryTxnBuilder
- Emailer
- ErrorLogger
- EventLogger
- GroovyServiceInvoker
- PropertyBuilder
- ReceiptSvc
- ServiceInvoker
- SvcConnUpdater
- TrackingCodeBuilder
- TxnBuilder
- TxnCheckpointSvc
- TxnUpdater
- UserBuilder

com.avoka.tm.test
- AbstractJUnitTest
- JUnitTestException
- JUnitTestRunner
- JUnitTestRunner.TestWrapper
- MockRegister
- MockRegistry
- MockRequest
- MockVoBuilder

com.avoka.tm.util
- Contract
- DeliveryResult
- DeliveryResultBuilder
- MemCache
- Path
- RedirectException
- Security
- Threads
- TxnUrlBuilder
- VelTemplate
- XmlDoc

com.avoka.tm.vo
- FileAttach
- Form
- Job
- JobAction
- JobStep
- Space
- SvcConn

- SvcDef
- Txn
- TxnCheckpoint
- User

Constants

com.avoka.core.groovy

Classes in Package com.avoka.core.groovy

- GroovyLogger

GroovyLogger

Package:

- com.avoka.core.groovy

Class GroovyLogger

- java.lang.Object
- com.avoka.core.groovy.GroovyLogger

```
public class GroovyLogger
extends Object
```

Provides a Groovy Logger class for logging information during the execution of Groovy scripts. Log message are logged to the server log file with the category `com.avoka.core.groovy.GroovyLogger`.

Example

The Groovy script example below imports the GroovyLogger class with the name 'logger' and then calls the debug, info, warn and error methods.

```
import com.avoka.core.groovy.GroovyLogger as logger

logger.debug 'this is a DEBUG level message'

logger.info 'this is an INFO level message'

logger.warn 'this is an WARN level message'

logger.error 'this is an ERROR level message'
```

This will result in the following logger messages being logged to the server log file. Note only WARN and ERROR level messages are logged by default.

```
 13:30:33,011 WARN  [com.avoka.core.groovy.GroovyLogger] (pool-5-thread-1) this is an WARN level message
 13:30:33,012 ERROR [com.avoka.core.groovy.GroovyLogger] (pool-5-thread-1) this is an ERROR level message
```

Note you can modify the standalone.xml configuration file to change category filtering level to show INFO and DEBUG level messages.

```
<logger category="com.avoka.core.groovy.GroovyLogger">
   <level name="WARN"/>
</logger>
```

Since:

- 4.3.0

Constructor Summary

Constructors

Constructor and Description
`GroovyLogger()`

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method and Description
`static void`	`clearTheadLogs()` Clear the thread logs.
`static void`	`debug(Object object)` Log the given DEBUG level message.
`static void`	`debug(String object)` Log the given DEBUG level message.

static void	error(Object object) Log the given ERROR level message.
static void	error(String object) Log the given ERROR level message.
static StringBuilder	getThreadDebugLevelLog() Return the thread local DEBUG level log.
static StringBuilder	getThreadInfoLevelLog() Return the thread local INFO level log.
static void	info(Object object) Log the given INFO level message.
static void	info(String object) Log the given INFO level message.
static void	setThreadDebugLevelLog(StringBuilder log) Set the thread local DEBUG level log.
static void	setThreadInfoLevelLog(StringBuilder log) Set the thread local INFO level log.
static void	warn(Object object) Log the given WARN level message.
static void	warn(String object) Log the given WARN level message.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

GroovyLogger

```
public GroovyLogger()
```

Method Detail

debug

```
public static void debug(Object object)
```

Log the given DEBUG level message.

Parameters:

- `object` - the DEBUG level message

---

debug

```
public static void debug(String object)
```

Log the given DEBUG level message.

Parameters:

- `object` - the DEBUG level message

---

error

```
public static void error(Object object)
```

Log the given ERROR level message.

Parameters:

- `object` - the ERROR level message

---

error

`public static void error(String object)`

Log the given ERROR level message.

Parameters:

- `object` - the ERROR level message

---

info

`public static void info(Object object)`

Log the given INFO level message.

Parameters:

- `object` - the INFO level message

---

info

`public static void info(String object)`

Log the given INFO level message.

Parameters:

- `object` - the INFO level message

---

warn

`public static void warn(Object object)`

Log the given WARN level message.

Parameters:

- `object` - the WARN level message

---

warn

`public static void warn(String object)`

Log the given WARN level message.

Parameters:

- `object` - the WARN level message

---

getThreadDebugLevelLog

`public static StringBuilder getThreadDebugLevelLog()`

Return the thread local DEBUG level log.

Returns:

- the thread local DEBUG level log

---

setThreadDebugLevelLog

`public static void setThreadDebugLevelLog(StringBuilder log)`

Set the thread local DEBUG level log.

Parameters:

- `log` - set the thread local DEBUG level log

---

getThreadInfoLevelLog

`public static StringBuilder getThreadInfoLevelLog()`

Return the thread local INFO level log.

Returns:

- the thread local INFO level log

---

setThreadInfoLevelLog

```
public static void setThreadInfoLevelLog(StringBuilder log)
```

Set the thread local INFO level log.

Parameters:

- `log` - set the INFO local debug level log

---

clearTheadLogs

```
public static void clearTheadLogs()
```

Clear the thread logs.

com.avoka.tm.http

Classes in Package com.avoka.tm.http

- DeleteRequest
- GetRequest
- HttpRequest
- HttpRequest.FileParam
- HttpRequest.Param
- HttpResponse
- PatchRequest
- PostRequest
- PutRequest
- RequestBuilder

DeleteRequest

Package:

- com.avoka.tm.http

Class DeleteRequest

- java.lang.Object
- com.avoka.tm.http.HttpRequest
- com.avoka.tm.http.DeleteRequest

```
public class DeleteRequest
extends HttpRequest
```

Provides a DELETE Request class for performing simple HTTP request operations. This class provides an easier and safer interface for the Apache HTTP Components library.

The default connection timeout is 10 seconds and socket read timeout is 60 seconds. Socket connections will also apply any JVM proxy settings automatically.

By default the maximum response read size is 16 MB. If the response is larger than this an IOException will be thrown. To increase the maximum read limit use the `HttpRequest.setReadLimit(int)` method.

In RESTful services the DELETE method is used to delete resources specified by the URI.

Examples

The example below performs DELETE request and checks the response status code.

```
import com.avoka.tm.http.*

String uri = 'https://service.mycorp.com/secure/rest/accounts/' + customerId

String username = svcDef.paramsMap.username
String password = svcDef.paramsMap.password

// execute DELETE request and return a HttpResponse object
HttpResponse response = new DeleteRequest(uri).setBasicAuth(username, password).execute()

// check HttpResponse status code
if (response.status == 200) {
   // Performed delete
   ...

} else if (response.status == 404) {
   // Not found
   ...

} else {
    throw new RuntimeException(response.statusLine)
}
```

Since:

- 5.0.0

See Also:

- GetRequest, PatchRequest, PutRequest, PostRequest

Nested Class Summary

Nested classes/interfaces inherited from class com.avoka.tm.http.HttpRequest

`HttpRequest.FileParam`, `HttpRequest.Param`

Constructor Summary

Constructors

Constructor and Description

`DeleteRequest(String uri)`

Create a DELETE HTTP request object with the given URI.

Method Summary

Methods inherited from class com.avoka.tm.http.HttpRequest

`addFileParam`, `addHeader`, `addHeaders`, `addParam`, `addParam`, `execute`, `getContext`, `getFileParams`, `getHeaders`, `getMessage`, `getMessageData`, `getMethod`, `getParams`, `getUri`, `setAcceptCompress`, `setBasicAuth`, `setCompressMessage`, `setConnectTimeout`, `setContentType`, `setContext`, `setMessage`, `setMessageData`, `setNtlmAuth`, `setParams`, `setProxy`, `setProxyAuth`, `setReadLimit`, `setSocketFactory`, `setSocketTimeout`, `setTimeouts`, `setUserAgent`, `toString`

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Constructor Detail

DeleteRequest

`public DeleteRequest(String uri)`

Create a DELETE HTTP request object with the given URI.

Parameters:

- `uri` - the request URI (required)

GetRequest

Package:

- com.avoka.tm.http

Class GetRequest

- java.lang.Object
- com.avoka.tm.http.HttpRequest
- com.avoka.tm.http.GetRequest

```
public class GetRequest
extends HttpRequest
```

Provides a GET Request class for performing simple HTTP request operations. This class provides an easier and safer interface for the Apache HTTP Components library.

The default connection timeout is 10 seconds and socket read timeout is 60 seconds. Socket connections will also apply any JVM proxy settings automatically.

By default the maximum response read size is 16 MB. If the response is larger than this an IOException will be thrown. To increase the maximum read limit use the `HttpRequest.setReadLimit(int)` method.

In RESTful services the GET method is used to retrieve resources specified by the URI.

Examples

The example below performs REST request and returns a JSON object.

```
import com.avoka.tm.http.*
import com.avoka.tm.util.*

Map params = [:]
params.id = request.getParameter('customerId')
params.account = request.getParameter('accountId')

String username = svcDef.paramsMap.username
String password = svcDef.paramsMap.password

// execute GET request and return a HttpResponse object
HttpResponse response = new GetRequest('https://service.mycorp.com/secure/rest/accounts/')
      .setParams(params)
      .setBasicAuth(username, password)
      .execute()

if (response.isStatusOK()) {
    // get Path object from the response
    Path path = response.getPathContent()
    ...

} else if (response.isStatusNotFound()) {
    // object not found
    ...

} else {
    throw new RuntimeException(response.statusLine)
}
```

Since:

- 5.0.0

See Also:

- DeleteRequest, PatchRequest, PutRequest, PostRequest

Nested Class Summary

Nested classes/interfaces inherited from class com.avoka.tm.http.HttpRequest

`HttpRequest.FileParam`, `HttpRequest.Param`

Constructor Summary

Constructors

Constructor and Description
`GetRequest(String uri)` Create a GET HTTP request object with the given URI.

Method Summary

Methods inherited from class com.avoka.tm.http.HttpRequest

`addFileParam`, `addHeader`, `addHeaders`, `addParam`, `addParam`, `execute`, `getContext`, `getFileParams`, `getHeaders`, `getMessage`, `getMessageData`, `getMethod`, `getParams`, `getUri`, `setAcceptCompress`, `setBasicAuth`, `setCompressMessage`, `setConnectTimeout`, `setContentType`, `setContext`, `setMessage`, `setMessageData`, `setNtlmAuth`, `setParams`, `setProxy`, `setProxyAuth`, `setReadLimit`, `setSocketFactory`, `setSocketTimeout`, `setTimeouts`, `setUserAgent`, `toString`

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Constructor Detail

GetRequest

```
public GetRequest(String uri)
```

Create a GET HTTP request object with the given URI.

Parameters:

- `uri` - the request URI (required)

HttpRequest

Package:

- com.avoka.tm.http

Class HttpRequest

- java.lang.Object
- com.avoka.tm.http.HttpRequest

Direct Known Subclasses:

- DeleteRequest, GetRequest, PatchRequest, PostRequest, PutRequest

---

```
public abstract class HttpRequest
extends Object
```

Provides an abstract HttpRequest class enabling simpler use of the Apache HTTP Components library.

The default connection timeout is 10 seconds and socket read timeout is 60 seconds. Socket connections will also apply any JVM proxy settings automatically.

By default the maximum response read size is 8 MB. If the response is larger than this an IOException will be thrown. To increase the maximum read limit use the `setReadLimit(int)` method.

Since:

- 5.0.0

See Also:

- DeleteRequest, GetRequest, PatchRequest, PostRequest, PutRequest

Nested Class Summary

Nested Classes

Modifier and Type	Class and Description
static class	HttpRequest.FileParam Provides a multi-part FileParam.
static class	HttpRequest.Param Provides a multi-part Param.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
HttpRequest	addFileParam(String paramName, byte[] fileData, String fileName) Add a file parameter for a multi-part request.
HttpRequest	addHeader(String name, String value) Add request header.
HttpRequest	addHeaders(Map<String,String> headers) Add all the request headers.
HttpRequest	addParam(String name, String value) Add a name and value request parameters.
HttpRequest	addParam(String name, String value, String contentType) Add a name and value request parameters.
HttpResponse	execute() Perform a HTTP request and return a response object.
HttpClientContext	getContext() Gets the HttpClientContext The context stores details about the call to a server.
Map<String,HttpRequest.FileParam>	getFileParams() Get file parameters for a multi-part request.

`Map<String,String>`	`getHeaders()` Return the request headers.
`String`	`getMessage()` Get the POST or PUT message body.
`byte[]`	`getMessageData()` Get the POST or PUT message body.
`com.avoka.tm.http.HttpRequest.Method`	`getMethod()` Return the request method.
`Map<String,HttpRequest.Param>`	`getParams()` Get the request form parameters.
`String`	`getUri()` Return the request URI.
`HttpRequest`	`setAcceptCompress(boolean acceptCompress)` Specify whether to set the header "Accept-Encoding: gzip, deflate".
`HttpRequest`	`setBasicAuth(String username, String password)` Set BASIC Authorization credentials.
`HttpRequest`	`setCompressMessage(boolean compress)` Specify whether to GZIP compress the POST or PUT message data.
`HttpRequest`	`setConnectTimeout(Integer connectTimeout)` Set the connection timeout in milliseconds.
`HttpRequest`	`setContentType(String contentType)` Set the 'Content-Type' header.
`HttpRequest`	`setContext(HttpClientContext context)` Sets the HttpClientContext The context stores details about the call to a server.
`HttpRequest`	`setMessage(String message)` Set the POST or PUT message body.
`HttpRequest`	`setMessageData(byte[] messageData)` Set the POST or PUT message body.
`HttpRequest`	`setNtlmAuth(String username, String password, String workstation, String domain)` Set NTLM Authorization credentials.
`HttpRequest`	`setParams(Map<String,String> params)` Set the request form parameters.
`HttpRequest`	`setProxy(String proxyHost, int proxyPort)` Set Proxy host and port.
`HttpRequest`	`setProxyAuth(String username, String password)` Set Proxy Authorization credentials.
`HttpRequest`	`setReadLimit(int readLimit)` Set the response content read limit in bytes.
`HttpRequest`	`setSocketFactory(LayeredConnectionSocketFactory socketFactory)` Set an optional ConnectionSocketFactory to enable creating layered sockets such as TLS.
`HttpRequest`	`setSocketTimeout(Integer socketTimeout)` Set the socket timeout in milliseconds.
`HttpRequest`	`setTimeouts(int connectTimeout, int socketTimeout)` Set the connect and socket timeouts in milliseconds.
`HttpRequest`	`setUserAgent(String userAgent)` Set the 'User-Agent' header.
`String`	`toString()` Return the string representation of this class.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Method Detail

setBasicAuth

```
public HttpRequest setBasicAuth(String username,
                                String password)
```

Set BASIC Authorization credentials.

Parameters:

- `username` - the user login name (required)
- `password` - the user login password (required)

Returns:

- the HttpRequest object

---

setNtlmAuth

```
public HttpRequest setNtlmAuth(String username,
                               String password,
                               String workstation,
                               String domain)
```

Set NTLM Authorization credentials.

Parameters:

- `username` - the user login name (required)
- `password` - the user login password (required)
- `workstation` - the workstation computer name (optional)
- `domain` - the name domain the user login belongs to (optional)

Returns:

- the HttpRequest object

---

setProxyAuth

```
public HttpRequest setProxyAuth(String username,
                                String password)
```

Set Proxy Authorization credentials.

Parameters:

- `username` - the proxy user login name (required)
- `password` - the proxy user login password (required)

Returns:

- the HttpRequest object

Since:

- 5.1.3

---

setProxy

```
public HttpRequest setProxy(String proxyHost,
                            int proxyPort)
```

Set Proxy host and port.

Parameters:

- `proxyHost` - proxy host
- `proxyPort` - proxy port

Returns:

- the HttpRequest object

Since:

- 5.1.3

---

getContext

```
public HttpClientContext getContext()
```

Gets the HttpClientContext The context stores details about the call to a server. It is useful in NTLM which has a fairly expensive authentication process. Once connected the authentication details are stored in the HttpClientContext and following HttpRequest calls do not need to be made. the process is as follows. Initially do a connection to a cheap service GetRequest. The execute() method creates the context. Create a new httpRequest with the expensive operation such as a PUT and call setContext(context)

Returns:

- the HttpClientContext context

---

setContext

```
public HttpRequest setContext(HttpClientContext context)
```

Sets the HttpClientContext The context stores details about the call to a server. It is useful in NTLM which has a fairly expensive authentication process. Once connected the authentication details are stored in the HttpClientContext and following HttpRequest calls do not need to be made. the process is as follows. Initially do a connection to a cheap service GetRequest. After the execute() method runs get the context. Create a new httpRequest with the expensive operation such as a PUT and call setContext(context)

Parameters:

- `context` - the HttpClientContext from a previous HttpRequest call.

Returns:

- the HttpRequest object

---

setContentType

```
public HttpRequest setContentType(String contentType)
```

Set the 'Content-Type' header.

Parameters:

- `contentType` - the 'Content-Type' header (required)

Returns:

- the HttpRequest object

---

getHeaders

```
public Map<String,String> getHeaders()
```

Return the request headers.

Returns:

- the request headers

Since:

- 5.1.7

---

getMethod

```
public com.avoka.tm.http.HttpRequest.Method getMethod()
```

Return the request method.

Returns:

- the request method

Since:

- 5.1.7

---

getUri

```
public String getUri()
```

Return the request URI.

Returns:

- the request URI

Since:

- 5.1.7

---

setUserAgent

```
public HttpRequest setUserAgent(String userAgent)
```

Set the 'User-Agent' header.

Parameters:

- `userAgent` - the 'User-Agent' header (required)

Returns:

- the HttpRequest object

---

addHeader

```
public HttpRequest addHeader(String name,
                             String value)
```

Add request header.

Parameters:

- `name` - the request header name (required)
- `value` - the request header value (required)

Returns:

- the HttpRequest object

---

addHeaders

```
public HttpRequest addHeaders(Map<String,String> headers)
```

Add all the request headers.

Parameters:

- `headers` - the map of request headers to add (required)

Returns:

- the HttpRequest object

---

setAcceptCompress

```
public HttpRequest setAcceptCompress(boolean acceptCompress)
```

Specify whether to set the header "Accept-Encoding: gzip, deflate".

Parameters:

- `acceptCompress` - specify whether to set the header "Accept-Encoding: gzip, deflate"

Returns:

- the HttpRequest object

---

setCompressMessage

```
public HttpRequest setCompressMessage(boolean compress)
```

Specify whether to GZIP compress the POST or PUT message data.

Parameters:

- `compress` - specify whether to GZIP compress the POST or PUT message data.

Returns:

- the HttpRequest object

---

setMessage

`public HttpRequest setMessage(String message)`

Set the POST or PUT message body. If specified the Content-Type will be set to 'plain/text'.

Parameters:

- `message` - the POST or PUT message body

Returns:

- the HttpRequest object

---

getMessage

`public String getMessage()`

Get the POST or PUT message body. If specified the Content-Type will be set to 'plain/text'.

Returns:

- the POST or PUT message body

Since:

- 5.1.7

---

setMessageData

`public HttpRequest setMessageData(byte[] messageData)`

Set the POST or PUT message body. If specified the Content-Type will be set to 'application/octet-stream'.

Parameters:

- `messageData` - the POST or PUT message body

Returns:

- the HttpRequest object

---

getMessageData

`public byte[] getMessageData()`

Get the POST or PUT message body. If specified the Content-Type will be set to 'application/octet-stream'.

Returns:

- the POST or PUT message body

Since:

- 5.1.7

---

setParams

`public HttpRequest setParams(Map<String,String> params)`

Set the request form parameters.

Parameters:

- `params` - the request form parameters

Returns:

- the HttpRequest object

---

getParams

`public Map<String,HttpRequest.Param> getParams()`

Get the request form parameters.

Returns:

- the request form parameters

Since:

- 5.1.7

---

addParam

```
public HttpRequest addParam(String name,
                            String value)
```

Add a name and value request parameters.

Parameters:

- `name` - the request parameter name (required)
- `value` - the request parameter value (required)

Returns:

- the HttpRequest object

---

addParam

```
public HttpRequest addParam(String name,
                            String value,
                            String contentType)
```

Add a name and value request parameters.

Parameters:

- `name` - the request parameter name (required)
- `value` - the request parameter value (required)
- `contentType` - the request parameter content type (required)

Returns:

- the HttpRequest object

Since:

- 5.0.1

---

addFileParam

```
public HttpRequest addFileParam(String paramName,
                                byte[] fileData,
                                String fileName)
```

Add a file parameter for a multi-part request.

Parameters:

- `paramName` - the request parameter name (required)
- `fileData` - the file data to add (required)
- `fileName` - the file data file name (required)

Returns:

- the HttpRequest object

---

getFileParams

```
public Map<String,HttpRequest.FileParam> getFileParams()
```

Get file parameters for a multi-part request.

Returns:

Since:

- 5.1.7

---

setTimeouts

```
public HttpRequest setTimeouts(int connectTimeout,
                               int socketTimeout)
```

Set the connect and socket timeouts in milliseconds.

Parameters:

- `connectTimeout` - the connection timeout in milliseconds

- `socketTimeout` - the socket timeout in milliseconds

Returns:

- the HttpRequest object

---

setSocketTimeout

`public HttpRequest setSocketTimeout(Integer socketTimeout)`

Set the socket timeout in milliseconds.

Parameters:

- `socketTimeout` - the socket timeout in milliseconds

Returns:

- the HttpRequest object

---

setConnectTimeout

`public HttpRequest setConnectTimeout(Integer connectTimeout)`

Set the connection timeout in milliseconds.

Parameters:

- `connectTimeout` - the connection timeout in milliseconds

Returns:

- the HttpRequest object

---

setReadLimit

`public HttpRequest setReadLimit(int readLimit)`

Set the response content read limit in bytes.

Parameters:

- `readLimit` - the response content read limit in bytes

Returns:

- the HttpRequest object

---

setSocketFactory

`public HttpRequest setSocketFactory(LayeredConnectionSocketFactory socketFactory)`

Set an optional ConnectionSocketFactory to enable creating layered sockets such as TLS. This method can be used to specify a `SSLConnectionSocketFactory` to support mutual SSL authentication.

Example

```
import java.io.File
import javax.net.ssl.SSLContext

import org.apache.http.ssl.SSLContexts
import org.apache.http.conn.ssl.TrustSelfSignedStrategy
import org.apache.http.conn.ssl.SSLConnectionSocketFactory

// Trust own CA and all self-signed certs
SSLContext sslContext = SSLContexts.custom()
    .loadTrustMaterial(new File("my.keystore"),
                       "nopassword".toCharArray(),
                       new TrustSelfSignedStrategy())
    .build()

// Allow TLSv1 protocol only
SSLConnectionSocketFactory socketFactory = new SSLConnectionSocketFactory(
    sslContext,
    new String[]{ "TLSv1" },
    null,
```

```
    SSLConnectionSocketFactory.getDefaultHostnameVerifier())

// execute GET request and return a HttpResponse object
def response = new GetRequest('https://service.mycorp.com/secure/rest/accounts/')
      .setBasicAuth(username, password)
      .setSocketFactory(socketFactory)
      .execute()
```

Parameters:

- `socketFactory` - the optional ConnectionSocketFactory used to build the client

Returns:

- the HttpRequest object

---

execute

```
public HttpResponse execute()
                    throws IOException
```

Perform a HTTP request and return a response object.

Returns:

- perform a HTTP request and return a response object

Throws:

- `IOException` - if an IO error occurs, please note HTTP status codes are returned in the response object

---

toString

```
public String toString()
```

Return the string representation of this class.

Overrides:

- `toString` in class `Object`

Returns:

- the string representation of this class

HttpRequest.FileParam

Package:

- com.avoka.tm.http

Class HttpRequest.FileParam

- java.lang.Object
- com.avoka.tm.http.HttpRequest.FileParam

Enclosing class:

- HttpRequest

```
public static class HttpRequest.FileParam
extends Object
```

Provides a multi-part FileParam.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`boolean`	`equals(Object obj)` Indicates whether some other object is "equal to" this one.
`int`	`hashCode()` Returns a hash code value for the object.
`String`	`toString()`

Methods inherited from class java.lang.Object

`getClass`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Method Detail

---

toString

```
public String toString()
```

Overrides:

- `toString` in class `Object`

---

hashCode

```
public int hashCode()
```

Returns a hash code value for the object. This method is supported for the benefit of hash tables such as those provided by `HashMap`. Implementation calculates hash code based on 'fileName' and 'fileData' fields with the help of `org.apache.commons.lang3.builder.HashCodeBuilder`

Overrides:

- `hashCode` in class `Object`

Returns:

- a hash code value for this object.

---

equals

```
public boolean equals(Object obj)
```

Indicates whether some other object is "equal to" this one. Implementation relies that the object is of same type and compares 'fileName' and 'fileData' with the help of `org.apache.commons.lang3.builder.EqualsBuilder` in order to return 'true'

Overrides:

- `equals` in class `Object`

Parameters:

- `obj` - an object.

Returns:

- `true` if this object is the same as the obj argument; `false` otherwise.

HttpRequest.Param

Package:

- com.avoka.tm.http

Class HttpRequest.Param

- java.lang.Object
- com.avoka.tm.http.HttpRequest.Param

Enclosing class:

- HttpRequest

```
public static class HttpRequest.Param
extends Object
```

Provides a multi-part Param.

Since:

- 5.0.1

Field Summary

Fields

Modifier and Type	Field and Description
`ContentType`	`contentType` Param Content Type.
`String`	`value` Param Value.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`boolean`	`equals(Object obj)` Indicates whether some other object is "equal to" this one.
`int`	`hashCode()` Returns a hash code value for the object.
`String`	`toString()`

Methods inherited from class java.lang.Object

`getClass`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Field Detail

---

value

```
public final String value
```

Param Value.

---

contentType

```
public final ContentType contentType
```

Param Content Type.

Method Detail

---

toString

```
public String toString()
```

Overrides:

- `toString` in class `Object`

---

hashCode

```
public int hashCode()
```

Returns a hash code value for the object. This method is supported for the benefit of hash tables such as those provided by `HashMap`. Implementation calculates hash code based on 'value' and 'contentType' fields with the help of `org.apache.commons.lang3.builder.HashCodeBuilder`

Overrides:

- `hashCode` in class `Object`

Returns:

- a hash code value for this object.

---

equals

```
public boolean equals(Object obj)
```

Indicates whether some other object is "equal to" this one. Implementation relies that the object is of same type and compares 'value' and 'contentType. toString()' with the help of `org.apache.commons.lang3.builder.EqualsBuilder` in order to return 'true'

Overrides:

- `equals` in class `Object`

Parameters:

- `obj` - an object.

Returns:

- `true` if this object is the same as the obj argument; `false` otherwise.

HttpResponse

Package:

- com.avoka.tm.http

Class HttpResponse

- java.lang.Object
- com.avoka.tm.http.HttpResponse

```
public class HttpResponse
extends Object
```

Provides a HttpResponse value object class.

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`HttpResponse()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`byte[]`	`getContent()`
`String`	`getContentEncoding()`
`long`	`getContentLength()`
`String`	`getContentType()`
`Document`	`getDocumentContent()` Return the content as a XML Document object, or null if not defined.
`Map<String,String>`	`getHeaders()` Return the map of HTTP response headers.
`Path`	`getPathContent()` Return the content as a Path object, or null if not defined.
`int`	`getStatus()`
`String`	`getStatusLine()`
`String`	`getTextContent()` Return the content as a UTF-8 encoded string, or null if not defined.
`boolean`	`isClientError()` Return true if HTTP response is an client error status code (400-499).
`boolean`	`isInformational()` Return true if HTTP response is an informational status code (100-199).
`boolean`	`isRedirection()` Return true if HTTP response is an redirection status code (300-399).
`boolean`	`isServerError()` Return true if HTTP response is an server error status code (500599).
`boolean`	`isStatusNotFound()` Return true if the HTTP response status code is NOT FOUND (404).
`boolean`	`isStatusOK()` Return true if the HTTP response status code is OK (200).
`boolean`	`isSuccess()`

	Return true if HTTP response is an success status code (200-299).
HttpResponse	setContent(byte[] content)
HttpResponse	setContentEncoding(String contentEncoding)
HttpResponse	setContentLength(long contentLength)
HttpResponse	setContentType(String contentType)
HttpResponse	setHeaders(Map<String,String> headers) Set the map of HTTP response headers.
HttpResponse	setStatus(int status)
HttpResponse	setStatusLine(String statusLine)
HttpResponse	setTextContent(String content) Sets the text content in the response.
String	toString() Return the string representation of this class.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

HttpResponse

```
public HttpResponse()
```

Method Detail

getPathContent

```
public Path getPathContent()
```

Return the content as a Path object, or null if not defined.

Returns:

- the content as Path object

---

getDocumentContent

```
public Document getDocumentContent()
```

Return the content as a XML Document object, or null if not defined.

Returns:

- the content as a XML Document object, or null if not defined.

---

getHeaders

```
public Map<String,String> getHeaders()
```

Return the map of HTTP response headers.

Returns:

- the map of HTTP response headers

---

setHeaders

```
public HttpResponse setHeaders(Map<String,String> headers)
```

Set the map of HTTP response headers.

Parameters:

- `headers` - the map of HTTP response headers

Returns:

- the HttpResponse value object

isStatusOK

```
public boolean isStatusOK()
```

Return true if the HTTP response status code is OK (200).

Returns:

- true if the HTTP response status code is OK (200).

---

isStatusNotFound

```
public boolean isStatusNotFound()
```

Return true if the HTTP response status code is NOT FOUND (404).

Returns:

- true if the HTTP response status code is NOT FOUND (404).

---

isInformational

```
public boolean isInformational()
```

Return true if HTTP response is an informational status code (100-199).

Returns:

- true if HTTP response is an informational status code (100-199).

---

isSuccess

```
public boolean isSuccess()
```

Return true if HTTP response is an success status code (200-299).

Returns:

- true if HTTP response is an success status code (200-299).

---

isRedirection

```
public boolean isRedirection()
```

Return true if HTTP response is an redirection status code (300-399).

Returns:

- true if HTTP response is an redirection status code (300-399).

---

isClientError

```
public boolean isClientError()
```

Return true if HTTP response is an client error status code (400-499).

Returns:

- true if HTTP response is an client error status code (400-499).

---

isServerError

```
public boolean isServerError()
```

Return true if HTTP response is an server error status code (500599).

Returns:

- true if HTTP response is an server error status code (500599).

---

getStatus

```
public int getStatus()
```

Returns:

- the HTTP response status code

---

setStatus

```
public HttpResponse setStatus(int status)
```

Parameters:

- `status` - the status to set

Returns:

- the HttpResponse value object

---

getStatusLine

```
public String getStatusLine()
```

Returns:

- the statusLine

---

setStatusLine

```
public HttpResponse setStatusLine(String statusLine)
```

Parameters:

- `statusLine` - the statusLine to set

Returns:

- the HttpResponse value object

---

getContent

```
public byte[] getContent()
```

Returns:

- the content

---

setContent

```
public HttpResponse setContent(byte[] content)
```

Parameters:

- `content` - the content to set

Returns:

- the HttpResponse value object

---

getContentLength

```
public long getContentLength()
```

Returns:

- the contentLength

---

setContentLength

```
public HttpResponse setContentLength(long contentLength)
```

Parameters:

- `contentLength` - the contentLength to set

Returns:

- the HttpResponse value object

---

getContentType

`public String getContentType()`

Returns:

- the contentType

---

setContentType

`public HttpResponse setContentType(String contentType)`

Parameters:

- `contentType` - the contentType to set

Returns:

- the HttpResponse value object

---

getContentEncoding

`public String getContentEncoding()`

Returns:

- the contentEncoding

---

setContentEncoding

`public HttpResponse setContentEncoding(String contentEncoding)`

Parameters:

- `contentEncoding` - the contentEncoding to set

Returns:

- the HttpResponse value object

---

getTextContent

`public String getTextContent()`

Return the content as a UTF-8 encoded string, or null if not defined.

Returns:

- the content as a UTF-8 encoded string, or null if not defined.

---

setTextContent

`public HttpResponse setTextContent(String content)`

Sets the text content in the response.

Parameters:

- `content` - the plain text content to set

Returns:

- the HttpResponse value object

Since:

- 5.1.7

---

toString

`public String toString()`

Return the string representation of this class.

Overrides:

- `toString` in class `Object`

Returns:

- the string representation of this class

PatchRequest

Package:

- com.avoka.tm.http

Class PatchRequest

- java.lang.Object
- com.avoka.tm.http.HttpRequest
- com.avoka.tm.http.PatchRequest

```
public class PatchRequest
extends HttpRequest
```

Provides a PATCH Request class for performing simple HTTP request operations. This class provides an easier and safer interface for the Apache HTTP Components library.

The default connection timeout is 10 seconds and socket read timeout is 60 seconds. Socket connections will also apply any JVM proxy settings automatically.

By default the maximum response read size is 16 MB. If the response is larger than this an IOException will be thrown. To increase the maximum read limit use the `HttpRequest.setReadLimit(int)` method.

In RESTful services the PATCH method is used to update resources.

Examples

The example below performs PUT request and converts the response into a JSON document.

```
import com.avoka.tm.http.*
import com.avoka.tm.util.*

String message = '''{
    "cutomerId":82881,
    "name":"John Doe",
    "email":"john.doe@gmail.com",
    "age":42
}'''

String username = svcDef.paramsMap.username
String password = svcDef.paramsMap.password

// execute PUT request and return a HttpResponse object
HttpResponse response = new PatchRequest('https://service.mycorp.com/secure/rest/accounts/')
      .setMessage(message)
      .setBasicAuth(username, password)
      .execute()

if (response.isStatusOK()) {
    // get JSON object from the response
    Path path = response.getPathContent()
    ...

} else if (response.isStatusNotFound()) {
    // object not found
    ...

} else {
    throw new RuntimeException(response.statusLine)
}
```

Since:

- 5.0.0

See Also:

- DeleteRequest, PutRequest, PostRequest

Nested Class Summary

Nested classes/interfaces inherited from class com.avoka.tm.http.HttpRequest

`HttpRequest.FileParam`, `HttpRequest.Param`

Constructor Summary

Constructors

Constructor and Description
`PatchRequest(String url)` Create a patch HTTP request object with the given URL.

Method Summary

Methods inherited from class com.avoka.tm.http.HttpRequest

`addFileParam`, `addHeader`, `addHeaders`, `addParam`, `addParam`, `execute`, `getContext`, `getFileParams`, `getHeaders`, `getMessage`, `getMessageData`, `getMethod`, `getParams`, `getUri`, `setAcceptCompress`, `setBasicAuth`, `setCompressMessage`, `setConnectTimeout`, `setContentType`, `setContext`, `setMessage`, `setMessageData`, `setNtlmAuth`, `setParams`, `setProxy`, `setProxyAuth`, `setReadLimit`, `setSocketFactory`, `setSocketTimeout`, `setTimeouts`, `setUserAgent`, `toString`

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Constructor Detail

PatchRequest

```
public PatchRequest(String url)
```

Create a patch HTTP request object with the given URL.

Parameters:

- `url` - the request URL (required)

PostRequest

Package:

- com.avoka.tm.http

Class PostRequest

- java.lang.Object
- com.avoka.tm.http.HttpRequest
- com.avoka.tm.http.PostRequest

```
public class PostRequest
extends HttpRequest
```

Provides a POST Request class for performing simple HTTP request operations. This class provides an easier and safer interface for the Apache HTTP Components library.

The default connection timeout is 10 seconds and socket read timeout is 60 seconds. Socket connections will also apply any JVM proxy settings automatically.

By default the maximum response read size is 16 MB. If the response is larger than this an IOException will be thrown. To increase the maximum read limit use the `HttpRequest.setReadLimit(int)` method.

In RESTful services the POST method is used to create resources (however its often used to perform updates as well).

Examples

The example below performs POST request and converts the response into a JSON document.

```
import com.avoka.tm.http.*
import com.avoka.tm.util.*

String message = '''{
    "name":"John Doe",
    "email":"john.doe@gmail.com",
    "age":42
}'''

String username = svcDef.paramsMap.username
String password = svcDef.paramsMap.password

// execute POST request and return a HttpResponse object
HttpResponse response = new PostRequest('https://service.mycorp.com/secure/rest/accounts/')
      .setMessage(message)
      .setBasicAuth(username, password)
      .execute()

// check HttpResponse status code
if (!response.isStatusOK()) {
    throw new RuntimeException(response.statusLine)
}

// get JSON text from the response
String json = response.getTextContent()
...
```

Since:

- 5.0.0

See Also:

- DeleteRequest, GetRequest, PatchRequest, PutRequest

Nested Class Summary

Nested classes/interfaces inherited from class com.avoka.tm.http.HttpRequest

`HttpRequest.FileParam`, `HttpRequest.Param`

Constructor Summary

Constructors

Constructor and Description
`PostRequest(String url)` Create a POST HTTP request object with the given URL.

Method Summary

Methods inherited from class com.avoka.tm.http.HttpRequest

`addFileParam`, `addHeader`, `addHeaders`, `addParam`, `addParam`, `execute`, `getContext`, `getFileParams`, `getHeaders`, `getMessage`, `getMessageData`, `getMethod`, `getParams`, `getUri`, `setAcceptCompress`, `setBasicAuth`, `setCompressMessage`, `setConnectTimeout`, `setContentType`, `setContext`, `setMessage`, `setMessageData`, `setNtlmAuth`, `setParams`, `setProxy`, `setProxyAuth`, `setReadLimit`, `setSocketFactory`, `setSocketTimeout`, `setTimeouts`, `setUserAgent`, `toString`

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Constructor Detail

PostRequest

```
public PostRequest(String url)
```

Create a POST HTTP request object with the given URL.

Parameters:

- `url` - the request URL (required)

PutRequest

Package:

- com.avoka.tm.http

Class PutRequest

- java.lang.Object
- com.avoka.tm.http.HttpRequest
- com.avoka.tm.http.PutRequest

```
public class PutRequest
extends HttpRequest
```

Provides a PUT Request class for performing simple HTTP request operations. This class provides an easier and safer interface for the Apache HTTP Components library.

The default connection timeout is 10 seconds and socket read timeout is 60 seconds. Socket connections will also apply any JVM proxy settings automatically.

By default the maximum response read size is 16 MB. If the response is larger than this an IOException will be thrown. To increase the maximum read limit use the `HttpRequest.setReadLimit(int)` method.

In RESTful services the PUT method is used to update resources.

Examples

The example below performs PUT request and converts the response into a JSON document.

```
import com.avoka.tm.http.*
import com.avoka.tm.util.*

String message = '''{
    "cutomerId":82881,
    "name":"John Doe",
    "email":"john.doe@gmail.com",
    "age":42
}'''

String username = svcDef.paramsMap.username
String password = svcDef.paramsMap.password

// execute PUT request and return a HttpResponse object
HttpResponse response = new PutRequest('https://service.mycorp.com/secure/rest/accounts/')
      .setMessage(message)
      .setBasicAuth(username, password)
      .execute()

if (response.isStatusOK()) {
    // get Path object from the response
    Path path = response.getPathContent()
    ...

} else if (response.isStatusNotFound()) {
    // object not found
    ...

} else {
    throw new RuntimeException(response.statusLine)
}
```

Since:

- 5.0.0

See Also:

- DeleteRequest, GetRequest, PatchRequest, PostRequest

Nested Class Summary

Nested classes/interfaces inherited from class com.avoka.tm.http.HttpRequest

`HttpRequest.FileParam`, `HttpRequest.Param`

Constructor Summary

Constructors

Constructor and Description
`PutRequest(String url)` Create a PUT HTTP request object with the given URL.

Method Summary

Methods inherited from class com.avoka.tm.http.HttpRequest

`addFileParam`, `addHeader`, `addHeaders`, `addParam`, `addParam`, `execute`, `getContext`, `getFileParams`, `getHeaders`, `getMessage`, `getMessageData`, `getMethod`, `getParams`, `getUri`, `setAcceptCompress`, `setBasicAuth`, `setCompressMessage`, `setConnectTimeout`, `setContentType`, `setContext`, `setMessage`, `setMessageData`, `setNtlmAuth`, `setParams`, `setProxy`, `setProxyAuth`, `setReadLimit`, `setSocketFactory`, `setSocketTimeout`, `setTimeouts`, `setUserAgent`, `toString`

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Constructor Detail

PutRequest

```
public PutRequest(String url)
```

Create a PUT HTTP request object with the given URL.

Parameters:

- `url` - the request URL (required)

RequestBuilder

Package:

- com.avoka.tm.http

Class RequestBuilder

- java.lang.Object
- com.avoka.tm.http.RequestBuilder

```
public class RequestBuilder
extends Object
```

Provides a HTTP Request Builder class.

Examples

Please find the Components HTTP Request Builder examples below.

Building Get Request Example

This Groovy example shows how to build GetRequest object

```
import com.avoka.tm.http.*
import com.avoka.tm.vo.*

SvcConn svcConn = svcDef.svcConn
String path = "https://service.mycorp.com/..."

GetRequest get = new RequestBuilder()
    .setSvcConn(svcConn)
    .setPath(path)
    .buildGet()
```

Building Post Request Example

This Groovy example shows how to build PostRequest object.

```
import com.avoka.tm.http.*
import com.avoka.tm.vo.*

SvcConn svcConn = svcDef.svcConn
String path = "https://service.mycorp.com/..."

PostRequest post = new RequestBuilder()
    .setSvcConn(svcConn)
    .setPath(path)
    .buildPost();
```

Since:

- 5.0.0

See Also:

- GetRequest, PostRequest

Constructor Summary

Constructors

Constructor and Description
`RequestBuilder()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
DeleteRequest	buildDelete() Return a new DeleteRequest object from the service connection parameter.
GetRequest	buildGet() Return a new GetRequest object from the service connection parameter.
PatchRequest	buildPatch() Return a new PatchRequest object from the service connection parameter.
PostRequest	buildPost() Return a new PostRequest object from the service connection parameter.
PutRequest	buildPut() Return a new PutRequest object from the service connection parameter.
RequestBuilder	setPath(String path) Set the path to append the to service connection endpoint for the request URL.
RequestBuilder	setSvcConn(SvcConn svcConn) Set the service connection parameter.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

RequestBuilder

```
public RequestBuilder()
```

Method Detail

setPath

```
public RequestBuilder setPath(String path)
```

Set the path to append the to service connection endpoint for the request URL.

Parameters:

- path - the to append the to service connection endpoint for the request URL

Returns:

- the request builder

---

setSvcConn

```
public RequestBuilder setSvcConn(SvcConn svcConn)
```

Set the service connection parameter.

Parameters:

- svcConn - the service connection parameter

Returns:

- the request builder

---

buildDelete

```
public DeleteRequest buildDelete()
```

Return a new DeleteRequest object from the service connection parameter.

Returns:

- a new DeleteRequest object from the service connection parameter

---

buildGet

```
public GetRequest buildGet()
```

Return a new GetRequest object from the service connection parameter.

Returns:

- a new GetRequest object from the service connection parameter

---

buildPatch

```
public PatchRequest buildPatch()
```

Return a new PatchRequest object from the service connection parameter.

Returns:

- a new PatchRequest object from the service connection parameter

---

buildPost

```
public PostRequest buildPost()
```

Return a new PostRequest object from the service connection parameter.

Returns:

- a new PostRequest object from the service connection parameter

---

buildPut

```
public PutRequest buildPut()
```

Return a new PutRequest object from the service connection parameter.

Returns:

- a new PutRequest object from the service connection parameter

com.avoka.tm.job

Classes in Package com.avoka.tm.job

- ActionResult
- ActionResultBuilder
- Jobs

ActionResult

Package:

- com.avoka.tm.job

Class ActionResult

- java.lang.Object
- com.avoka.fc.core.service.job.ActionResult
- com.avoka.tm.job.ActionResult

```
public class ActionResult
extends com.avoka.fc.core.service.job.ActionResult
```

Provides a Job Action Result class.

Since:

- 5.0.0

Nested Class Summary

Nested classes/interfaces inherited from class com.avoka.fc.core.service.job.ActionResult

`com.avoka.fc.core.service.job.ActionResult.Status`

Constructor Summary

Constructors

Constructor and Description
`ActionResult(com.avoka.fc.core.service.job.ActionResult.Status status)` Create a new Job Action Result object with the given status.

Method Summary

Methods inherited from class com.avoka.fc.core.service.job.ActionResult

`getExpiryTime, getMaxRetryAttempts, getMaxRetryAttepts, getMessage, getNextRetryAttempt, getRoute, getStatus, isStatusError, setExpiryTime, setMaxRetryAttempts, setMaxRetryAttepts, setMessage, setNextRetryAttempt, setRoute, toActionStatus, toString`

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

ActionResult

```
public ActionResult(com.avoka.fc.core.service.job.ActionResult.Status status)
```

Create a new Job Action Result object with the given status.

Parameters:

- `status` - the result status

ActionResultBuilder

Package:

- com.avoka.tm.job

Class ActionResultBuilder

- java.lang.Object
- com.avoka.tm.job.ActionResultBuilder

```
public class ActionResultBuilder
extends Object
```

Provides a job action invocation ActionResult builder.

Examples

The example below uses creates a ActionResult object to signal the job step action has successfully completed.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.job.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentJobActionService {

      ActionResult invoke(SvcDef svcDef, Job job, JobAction jobAction, HttpServletRequest request, User user) {

          // Business logic ...

          String xml = Jobs.getStartTxnXml(job)
          Path path = new Path(xml)
          String refNumber = path.val('//Application/RefNumber')

          return new ActionResultBuilder()
              .setStatus('Completed')
              .setMessage('Action completed for: ' + refNumber)
              .build()
      }
}
```

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`ActionResultBuilder()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`ActionResult`	`build()` Return a new job action result object based on the specified properties.
`ActionResultBuilder`	`setExpiryTime(Date expiryTime)` Set the step action expiry time after which the job action should be cancelled.
`ActionResultBuilder`	`setMaxRetryAttempts(Integer maxRetryAttempts)` Set the maximum number of time to attempt performing the step action.
`ActionResultBuilder`	`setMessage(String message)`

	Set the job action result message for process auditing purposes.
ActionResultBuilder	setNextRetryAttempt(Date nextRetryAttempt) Set the time to next retry performing step action.
ActionResultBuilder	setNextRetryAttemptMins(int minutes) The next time in minutes to retry performing the action.
ActionResultBuilder	setRoute(String route) Set the job action result route step.
ActionResultBuilder	setStatus(String status) Set the job action result status.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ActionResultBuilder

```
public ActionResultBuilder()
```

Method Detail

setStatus

```
public ActionResultBuilder setStatus(String status)
```

Set the job action result status.

Parameters:

- `status` - the job action result status

Returns:

- the job action result builder

---

setRoute

```
public ActionResultBuilder setRoute(String route)
```

Set the job action result route step.

Parameters:

- `route` - the step name to route the job flow to

Returns:

- the job action result builder

---

setMessage

```
public ActionResultBuilder setMessage(String message)
```

Set the job action result message for process auditing purposes.

Parameters:

- `message` - the action result message for process auditing purposes

Returns:

- the job action result builder

---

setExpiryTime

```
public ActionResultBuilder setExpiryTime(Date expiryTime)
```

Set the step action expiry time after which the job action should be cancelled.

Parameters:

- `expiryTime` - the step action expiry time after which the job action should be cancelled

Returns:

- the job action result builder

---

setMaxRetryAttempts

`public ActionResultBuilder setMaxRetryAttempts(Integer maxRetryAttempts)`

Set the maximum number of time to attempt performing the step action.

Parameters:

- `maxRetryAttempts` - the maximum number of time to attempt performing the step action

Returns:

- the job action result builder

---

setNextRetryAttempt

`public ActionResultBuilder setNextRetryAttempt(Date nextRetryAttempt)`

Set the time to next retry performing step action.

Parameters:

- `nextRetryAttempt` - the time to next retry performing step action

Returns:

- the job action result builder

---

setNextRetryAttemptMins

`public ActionResultBuilder setNextRetryAttemptMins(int minutes)`

The next time in minutes to retry performing the action.

Parameters:

- `minutes` - the next time in minutes to retry performing the action

Returns:

- the job action result builder

---

build

`public ActionResult build()`

Return a new job action result object based on the specified properties.

Returns:

- a new job action result object based on the specified properties.

Jobs

Package:

- com.avoka.tm.job

Class Jobs

- java.lang.Object
- com.avoka.tm.job.Jobs

```
public class Jobs
extends Object
```

Provides Job utility functions.

Examples

Get Start Transaction

Return the start (first) transaction for the job.

```
import com.avoka.tm.job.*
import com.avoka.tm.vo.*

Txn txn = Jobs.getStartTxn(job)
```

Get Start Transaction XML Data

Return the XML of the start transaction for the job.

```
import com.avoka.tm.job.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

String xml = Jobs.getStartTxnXml(job)

Path path = new Path(xml)
```

Get Start User

Return the User of the starting transaction for the job.

```
import com.avoka.tm.job.*
import com.avoka.tm.vo.*

User user = Jobs.getStartUser(job)
```

Since:

- 5.0.0

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method and Description
`static Txn`	`getStartTxn(Job job)` Return the start transaction for the given job.
`static String`	`getStartTxnXml(Job job)` Return the start transaction XML for the given job.
`static User`	`getStartUser(Job job)`

	Return the start user for the given job.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Method Detail

getStartTxn

```
public static Txn getStartTxn(Job job)
```

Return the start transaction for the given job. The returned transaction include formDataMap, groupNames and propertyMap.

Example

```
import com.avoka.tm.job.*
import com.avoka.tm.vo.*

Txn txn = Jobs.getStartTxn(job)
```

Parameters:

- `job` - the job to query (required)

Returns:

- the start transaction, or null if not found

---

getStartTxnXml

```
public static String getStartTxnXml(Job job)
```

Return the start transaction XML for the given job.

Example

```
import com.avoka.tm.job.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

String xml = Jobs.getStartTxnXml(job)

Path path = new Path(xml)
```

Parameters:

- `job` - the job to query (required)

Returns:

- the start transaction XML, or null if not found

---

getStartUser

```
public static User getStartUser(Job job)
```

Return the start user for the given job.

Example

```
import com.avoka.tm.job.*
import com.avoka.tm.vo.*

User user = Jobs.getStartUser(job)
```

Parameters:

- `job` - the job to query (required)

Returns:

- the start user, or null if not found

com.avoka.tm.query

Classes in Package com.avoka.tm.query

- JobQuery
- PropertyQuery
- SpaceQuery
- SvcConnQuery
- SvcDefQuery
- TxnQuery
- UserQuery

JobQuery

Package:

- com.avoka.tm.query

Class JobQuery

- java.lang.Object
- com.avoka.tm.query.JobQuery

```
public class JobQuery
extends Object
```

Provides a job value object query class.

Examples

Please find the job query examples about list, get first and count below.

Job Query List Example

This Groovy example shows how to list jobs matching certain criteria ordered descending by id (maximum number of 4 records).

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

String jobName = "job name..."

List<Job> jobs = new JobQuery()
    .setName(jobName)
    .setStatus(Job.STATUS_IN_PROGRESS)
    .addOrderByDesc("timeLastProcessed")
    .setFetchLimit(100)
    .listValues()

// In JSON format
String jobsJson = new JobQuery()
    .setName(jobName)
    .setStatus(Job.STATUS_IN_PROGRESS)
    .addOrderBy("timeLastProcessed", true)
    .setFetchLimit(100)
    .listJson()
```

Job Query First Example

This Groovy example shows how to get first job matching certain criteria ordered descending by id.

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

Job job = new JobQuery()
    .setName(jobName)
    .addOrderByDesc("id")
    .firstValue()

// In JSON format
String jobJson = new JobQuery()
    .setName(jobName)
    .setStatus(Job.STATUS_COMPLETED)
    .firstJson()
```

Job Query Count Example

This Groovy example shows how to count all jobs matching certain criteria.

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

int jobInStatusCount = new JobQuery()
    .setStatus(Job.STATUS_IN_PROGRESS)
    .count()
```

Since:

- 5.0.0

See Also:

- `Job`

Constructor Summary

Constructors

Constructor and Description
`JobQuery()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`JobQuery`	`addOrderByAsc(String orderProperty)` Add the sort order by ascending to the query.
`JobQuery`	`addOrderByDesc(String orderProperty)` Add the sort order by descending to the query.
`int`	`count()` Execute a select count query and return the total number of records selected by the query.
`String`	`firstJson()` Execute the query and return the first job JSON value.
`Job`	`firstValue()` Execute the query and return the first job value object for the query.
`String`	`listJson()` Execute the query and return an jobs JSON array list.
`List<Job>`	`listValues()` Execute the job query and return a list of Job value objects.
`JobQuery`	`setFetchLimit(int fetchLimit)` Set the query fetch limit to limit the maximum number of records returned.
`JobQuery`	`setId(Long id)` Set the job id (PK) query parameter.
`JobQuery`	`setJobKey(String jobKey)` Set the job key query parameter.
`JobQuery`	`setName(String name)` Set the job name query parameter.
`JobQuery`	`setReferenceNumber(String referenceNumber)` Set the job reference number query parameter.
`JobQuery`	`setStatus(String status)` Set the job status query parameter.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

JobQuery

```
public JobQuery()
```

Method Detail

setId

```
public JobQuery setId(Long id)
```

Set the job id (PK) query parameter.

Parameters:

- `id` - the job id (PK) query parameter

Returns:

- the job query

---

setName

```
public JobQuery setName(String name)
```

Set the job name query parameter.

Parameters:

- `name` - the job name query parameter

Returns:

- the job query

---

setJobKey

```
public JobQuery setJobKey(String jobKey)
```

Set the job key query parameter.

Parameters:

- `jobKey` - the job key query parameter

Returns:

- the job query

---

setStatus

```
public JobQuery setStatus(String status)
```

Set the job status query parameter.

Parameters:

- `status` - the job status query parameter

Returns:

- the job query

---

setReferenceNumber

```
public JobQuery setReferenceNumber(String referenceNumber)
```

Set the job reference number query parameter.

Parameters:

- `referenceNumber` - the job reference number query parameter

Returns:

- the job query

---

setFetchLimit

```
public JobQuery setFetchLimit(int fetchLimit)
```

Set the query fetch limit to limit the maximum number of records returned. The default query fetch limit is 100 records.

Parameters:

- `fetchLimit` - the query fetch limit

Returns:

- the job query

---

addOrderByAsc

```
public JobQuery addOrderByAsc(String orderProperty)
```

Add the sort order by ascending to the query.

Parameters:

- `orderProperty` - the property to sort by (required)

Returns:

- the job query

---

addOrderByDesc

```
public JobQuery addOrderByDesc(String orderProperty)
```

Add the sort order by descending to the query.

Parameters:

- `orderProperty` - the property to sort by (required)

Returns:

- the job query

---

count

```
public int count()
```

Execute a select count query and return the total number of records selected by the query.

Returns:

- the total number of records selected by the query

---

listValues

```
public List<Job> listValues()
```

Execute the job query and return a list of Job value objects.

Returns:

- execute the job query and return a list of Job value objects

---

listJson

```
public String listJson()
```

Execute the query and return an jobs JSON array list.

Returns:

- execute the query and return an jobs JSON array list

---

firstValue

```
public Job firstValue()
```

Execute the query and return the first job value object for the query.

Returns:

- execute the query and return the first job value object for the query

---

firstJson

```
public String firstJson()
```

Execute the query and return the first job JSON value.

Returns:

- execute the query and return the first job JSON value

PropertyQuery

Package:

- com.avoka.tm.query

Class PropertyQuery

- java.lang.Object
- com.avoka.tm.query.PropertyQuery

```
public class PropertyQuery
extends Object
```

Provides a property value query service.

Property values are cached in memory for a period of 30 seconds to improve database performance.

Examples

Example scripts are provided below.

Form Property Example

The example below get the "Product Codes" form version property value for the form's current form version. The property value may be use in Form Prefill or Dynamic Data services.

```
import com.avoka.tm.query.*

String value = new PropertyQuery()
    .setName("Product Codes")
    .setFormCode("PDS-12")
    .getValue()
```

Organization Property Example

The example below will get the "Loan Types" Organization property value.

```
import com.avoka.tm.query.*

String value = new PropertyQuery()
    .setName("Loan Types")
    .setClientCode("maguire")
    .getValue()
```

Form Space Property Example

The example below will get the "Locale Messages" Form Space property value.

```
import com.avoka.tm.query.*

String value = new PropertyQuery()
    .setName("Locale Messages")
    .setSpaceName("Work Space")
    .getValue()
```

Transaction Property Lookup Example

The example below will get the "Saved Email" via the specified Txn value object. This method will attempt to resolve the property value in the order:

Form Version Property Value Client Property Value Portal Property Value Deployment Property Value

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*
```

```
Txn txn = ...

String value = new PropertyQuery()
    .setName("Saved Email")
    .setTxn(txn)
    .getValue()
```

Cached Organization Reference Data Example

The example below will get the "maguire" organization's "Product Codes" reference data. This query data will be cache the value in memory for a period 60 minutes.

```
import com.avoka.tm.query.*

String refData = new PropertyQuery()
    .setName("Product Codes")
    .setClientCode("maguire")
    .setCacheTimeout(60)
    .getValue()
```

Since:

- 5.0.0

See Also:

- MemCache

Constructor Summary

Constructors

Constructor and Description
`PropertyQuery()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`String`	`getValue()` Return the property value for the query parameters.
`PropertyQuery`	`setCacheTimeout(int timeoutMins)` Set the property query cache timeout in minutes.
`PropertyQuery`	`setClientCode(String clientCode)` Set the organization client code parameter.
`PropertyQuery`	`setFormCode(String formCode)` Set the form code parameter.
`PropertyQuery`	`setName(String name)` Set the property name parameter.
`PropertyQuery`	`setSpaceName(String spaceName)` Set the property space name parameter.
`PropertyQuery`	`setTxn(Txn txn)` Set the transaction to resolve the property value against.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

PropertyQuery

```
public PropertyQuery()
```

Method Detail

setName

`public PropertyQuery setName(String name)`

Set the property name parameter.

Parameters:

- `name` - the property name

Returns:

- the property query

---

setFormCode

`public PropertyQuery setFormCode(String formCode)`

Set the form code parameter.

Parameters:

- `formCode` - the property form code

Returns:

- the property query

---

setClientCode

`public PropertyQuery setClientCode(String clientCode)`

Set the organization client code parameter.

Parameters:

- `clientCode` - the property client code

Returns:

- the property query

---

setSpaceName

`public PropertyQuery setSpaceName(String spaceName)`

Set the property space name parameter.

Parameters:

- `spaceName` - the property space name

Returns:

- the property query

---

setTxn

`public PropertyQuery setTxn(Txn txn)`

Set the transaction to resolve the property value against.

Parameters:

- `txn` - the transaction to resolve the property against

Returns:

- the property query

---

setCacheTimeout

`public PropertyQuery setCacheTimeout(int timeoutMins)`

Set the property query cache timeout in minutes.

Parameters:

- `timeoutMins` - the query cache timeout in minutes, must be a positive value.

Returns:

- the property query

---

getValue

```
public String getValue()
```

Return the property value for the query parameters.

Returns:

- the property value for the query parameter

SpaceQuery

Package:

- com.avoka.tm.query

Class SpaceQuery

- java.lang.Object
- com.avoka.tm.query.SpaceQuery

```
public class SpaceQuery
extends Object
```

Provides a form space query class.

Examples

Please find the space query examples about list, get first and count below.

Space Query List Example

This Groovy example shows how to list spaces matching certain criteria ordered ascending by name (maximum number of 4 records).

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

List<Space> spaces = new SpaceQuery()
    .setFetchLimit(4)
    .addOrderByAsc("name")
    .listValues()

// In JSON format
String spacesJson = new SpaceQuery()
    .setFetchLimit(4)
    .addOrderByAsc("name")
    .listJson()
```

Space Query First Example

This Groovy example shows how to get first space matching certain criteria ordered descending by id.

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

Space space = new SpaceQuery()
    .setName(spaceName)
    .firstValue()

// In JSON format
String spaceJson = new SpaceQuery()
    .setName(spaceName)
    .firstJson()
```

Space Query Count Example

This Groovy example shows how to count all spaces matching certain criteria.

```
import com.avoka.tm.query.*

int spaceWithCtxPathCount = new SpaceQuery()
    .setContextPath(ctxPath)
    .count()
```

Since:

- 5.0.0

See Also:

- Space

Constructor Summary

Constructors

Constructor and Description
`SpaceQuery()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`SpaceQuery`	`addOrderByAsc(String orderProperty)` Add the sort order by ascending to the query.
`SpaceQuery`	`addOrderByDesc(String orderProperty)` Add the sort order by descending to the query.
`int`	`count()` Execute a select count query and return the total number of records selected by the query.
`String`	`firstJson()` Execute the query and return the first space JSON value.
`Space`	`firstValue()` Execute the query and return the first space value object for the query.
`String`	`listJson()` Execute the query and return a space JSON array list.
`List<Space>`	`listValues()` Execute the space query and return a list of Space value objects.
`SpaceQuery`	`setContextPath(String contextPath)` Set the space context path parameter.
`SpaceQuery`	`setFetchLimit(int fetchLimit)` Set the query fetch limit to limit the maximum number of records returned.
`SpaceQuery`	`setId(Long id)` Set the space id (PK) query parameter.
`SpaceQuery`	`setName(String name)` Set the space name query parameter.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

SpaceQuery

```
public SpaceQuery()
```

Method Detail

setId

```
public SpaceQuery setId(Long id)
```

Set the space id (PK) query parameter.

Parameters:

- `id` - the space id (PK) query parameter

Returns:

- the space query

---

setName

`public SpaceQuery setName(String name)`

Set the space name query parameter.

Parameters:

- `name` - the space name query parameter

Returns:

- the space query

---

setContextPath

`public SpaceQuery setContextPath(String contextPath)`

Set the space context path parameter.

Parameters:

- `contextPath` - the space content path parameter

Returns:

- the space query

---

setFetchLimit

`public SpaceQuery setFetchLimit(int fetchLimit)`

Set the query fetch limit to limit the maximum number of records returned. The default query fetch limit is 100 records.

Parameters:

- `fetchLimit` - the query fetch limit

Returns:

- the space query

---

addOrderByAsc

`public SpaceQuery addOrderByAsc(String orderProperty)`

Add the sort order by ascending to the query.

Parameters:

- `orderProperty` - the property to sort by (required)

Returns:

- the space query

---

addOrderByDesc

`public SpaceQuery addOrderByDesc(String orderProperty)`

Add the sort order by descending to the query.

Parameters:

- `orderProperty` - the property to sort by (required)

Returns:

- the space query

---

count

`public int count()`

Execute a select count query and return the total number of records selected by the query.

Returns:

- the total number of records selected by the query

---

listValues

```
public List<Space> listValues()
```

Execute the space query and return a list of Space value objects.

Returns:

- execute the space query and return a list of Space value objects

---

listJson

```
public String listJson()
```

Execute the query and return a space JSON array list.

Returns:

- execute the query and return a space JSON array list

---

firstValue

```
public Space firstValue()
```

Execute the query and return the first space value object for the query.

Returns:

- execute the query and return the first space value object for the query

---

firstJson

```
public String firstJson()
```

Execute the query and return the first space JSON value.

Returns:

- execute the query and return the first space JSON value

SvcConnQuery

Package:

- com.avoka.tm.query

Class SvcConnQuery

- java.lang.Object
- com.avoka.tm.query.SvcConnQuery

```
public class SvcConnQuery
extends Object
```

Provides a service connection value object query class.

Examples

Please find the service connection query examples about list and first and count below.

Service Connection Query List Example

This Groovy example shows how to list "AWS S3" type connections ordered by name.

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

List<SvcConn> svcConns = new SvcConnQuery()
    .setType("AWS S3")
    .addOrderByAsc("name")
    .listValues()
```

Service Connection Query First Example

This Groovy example shows how to query a service connection by name.

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

SvcConn svcConn = new SvcConnQuery()
    .setName("History Publish")
    .setClientCode("clientCode")
    .firstValue();
```

Service Connection Query Count Example

This Groovy example shows how to count service connection with given name.

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

int count = new SvcConnQuery()
    .setName("IDV Conn")
    .count();
```

Since:

- 5.1.0

Constructor Summary

Constructors

Constructor and Description

SvcConnQuery()

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
SvcConnQuery	addOrderByAsc(String orderProperty) Add the sort order by ascending to the query.
SvcConnQuery	addOrderByDesc(String orderProperty) Add the sort order by descending to the query.
int	count() Execute a select count query and return the total number of records selected by the query.
String	firstJson() Execute the query and return the first JSON value.
SvcConn	firstValue() Execute the query and return the first value object for the query.
String	listJson() Execute the query and return JSON array list.
List<SvcConn>	listValues() Execute the query and return a list of value objects.
SvcConnQuery	setClientCode(String clientCode) Set the service connection client code query parameter.
SvcConnQuery	setFetchLimit(int fetchLimit) Set the query fetch limit to limit the maximum number of records returned.
SvcConnQuery	setId(Long id) Set the service connection id (PK) query parameter.
SvcConnQuery	setName(String name) Set the service connection name query parameter.
SvcConnQuery	setType(String type) Set the service connection type query parameter.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SvcConnQuery

```
public SvcConnQuery()
```

Method Detail

setClientCode

```
public SvcConnQuery setClientCode(String clientCode)
```

Set the service connection client code query parameter.

Parameters:

- clientCode - the service connection client code query parameter

Returns:

- the query object

setId

```
public SvcConnQuery setId(Long id)
```

Set the service connection id (PK) query parameter.

Parameters:

- `id` - the service connection (PK) query parameter

Returns:

- the query object

---

setName

`public SvcConnQuery setName(String name)`

Set the service connection name query parameter.

Parameters:

- `name` - the service connection name query parameter

Returns:

- the query object

---

setType

`public SvcConnQuery setType(String type)`

Set the service connection type query parameter.

Parameters:

- `type` - the service connection type query parameter

Returns:

- the query object

---

setFetchLimit

`public SvcConnQuery setFetchLimit(int fetchLimit)`

Set the query fetch limit to limit the maximum number of records returned. The default query fetch limit is 100 records.

Parameters:

- `fetchLimit` - the query fetch limit

Returns:

- the query object

---

addOrderByAsc

`public SvcConnQuery addOrderByAsc(String orderProperty)`

Add the sort order by ascending to the query.

Parameters:

- `orderProperty` - the property to sort by (required)

Returns:

- the query object

---

addOrderByDesc

`public SvcConnQuery addOrderByDesc(String orderProperty)`

Add the sort order by descending to the query.

Parameters:

- `orderProperty` - the property to sort by (required)

Returns:

- the query object

---

count

```
public int count()
```

Execute a select count query and return the total number of records selected by the query.

Returns:

- the total number of records selected by the query

---

listValues

```
public List<SvcConn> listValues()
```

Execute the query and return a list of value objects.

Returns:

- execute the query and return a list of value objects

---

listJson

```
public String listJson()
```

Execute the query and return JSON array list.

Returns:

- execute the query and return JSON array list

---

firstValue

```
public SvcConn firstValue()
```

Execute the query and return the first value object for the query.

Returns:

- execute the query and return the first value object for the query

---

firstJson

```
public String firstJson()
```

Execute the query and return the first JSON value.

Returns:

- execute the query and return the first JSON value

SvcDefQuery

Package:

- com.avoka.tm.query

Class SvcDefQuery

- java.lang.Object
- com.avoka.tm.query.SvcDefQuery

```
public class SvcDefQuery
extends Object
```

Provides a service definition value object query class.

Examples

Please find the service definition query examples.

Service Definition List Example

This Groovy example shows how to list service definitions matching.

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

List<SvcDef> svcDefs = new SvcDefQuery()
     .setType("Dynamic Data")
     .withCurrentVersion()
     .listValues()

// In JSON format
String svcDefListJson = new SvcDefQuery()
     .setName("myServiceDefName")
     .listJson()
```

Since:

- 5.1.0

See Also:

- SvcDef

Constructor Summary

Constructors

Constructor and Description
`SvcDefQuery()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`SvcDefQuery`	`addOrderByAsc(String orderProperty)` Add the sort order by ascending to the query.
`SvcDefQuery`	`addOrderByDesc(String orderProperty)` Add the sort order by descending to the query.
`int`	`count()` Execute a select count query and return the total number of records selected by the query.
`String`	`firstJson()` Execute the query and return the first service definition JSON value.

SvcDef	firstValue() Execute the query and return the first service definition value object for the query.
String	listJson() Execute the query and return service definition JSON array list.
List<SvcDef>	listValues() Execute the service definition query and return a list of SvcDef value objects.
SvcDefQuery	setClientCode(String clientCode) Set the service definition client code parameter.
SvcDefQuery	setCreatedBy(String createdBy) Set the service definition created by parameter.
SvcDefQuery	setFetchLimit(int fetchLimit) Set the query fetch limit to limit the maximum number of records returned.
SvcDefQuery	setId(Number id) Set the service definition id (PK) query parameter.
SvcDefQuery	setLastModifiedBy(String lastModifiedBy) Set the service definition last modified by parameter.
SvcDefQuery	setName(String name) Set the service definition name parameter.
SvcDefQuery	setType(String type) Set the service definition type parameter.
SvcDefQuery	setVersionNumber(int versionNumber) Set the service definition version number parameter.
SvcDefQuery	withCurrentVersion() Set the query to return the service definition of current version.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SvcDefQuery

```
public SvcDefQuery()
```

Method Detail

setId

```
public SvcDefQuery setId(Number id)
```

Set the service definition id (PK) query parameter.

Parameters:

- id - the service definition id (PK) query parameter

Returns:

- the service definition query

---

setVersionNumber

```
public SvcDefQuery setVersionNumber(int versionNumber)
```

Set the service definition version number parameter.

Parameters:

- versionNumber - the service definition version number parameter

Returns:

- the service definition query

---

setName

`public SvcDefQuery setName(String name)`

Set the service definition name parameter.

Parameters:

- `name` - the service definition name parameter

Returns:

- the service definition query

---

setClientCode

`public SvcDefQuery setClientCode(String clientCode)`

Set the service definition client code parameter.

Parameters:

- `clientCode` - the service definition client code parameter

Returns:

- the service definition query

---

setType

`public SvcDefQuery setType(String type)`

Set the service definition type parameter.

Parameters:

- `type` - the service definition type parameter

Returns:

- the service definition query

---

setCreatedBy

`public SvcDefQuery setCreatedBy(String createdBy)`

Set the service definition created by parameter.

Parameters:

- `createdBy` - the service definition created by parameter

Returns:

- the service definition query

---

setLastModifiedBy

`public SvcDefQuery setLastModifiedBy(String lastModifiedBy)`

Set the service definition last modified by parameter.

Parameters:

- `lastModifiedBy` - the service definition created by parameter

Returns:

- the service definition query

---

withCurrentVersion

`public SvcDefQuery withCurrentVersion()`

Set the query to return the service definition of current version.

Returns:

- the service definition query

addOrderByAsc

```
public SvcDefQuery addOrderByAsc(String orderProperty)
```

Add the sort order by ascending to the query.

Parameters:

- `orderProperty` - the property to sort by (required)

Returns:

- the service definition query

---

addOrderByDesc

```
public SvcDefQuery addOrderByDesc(String orderProperty)
```

Add the sort order by descending to the query.

Parameters:

- `orderProperty` - the property to sort by (required)

Returns:

- the service definition query

---

setFetchLimit

```
public SvcDefQuery setFetchLimit(int fetchLimit)
```

Set the query fetch limit to limit the maximum number of records returned. The default query fetch limit is 100 records.

Parameters:

- `fetchLimit` - the query fetch limit

Returns:

- the service definition query

---

count

```
public int count()
```

Execute a select count query and return the total number of records selected by the query.

Returns:

- the total number of records selected by the query

---

listValues

```
public List<SvcDef> listValues()
```

Execute the service definition query and return a list of SvcDef value objects.

Returns:

- execute the service definition query and return a list of SvcDef value objects

---

listJson

```
public String listJson()
```

Execute the query and return service definition JSON array list.

Returns:

- execute the query and return service definition JSON array list

---

firstValue

```
public SvcDef firstValue()
```

Execute the query and return the first service definition value object for the query.

Returns:

- execute the query and return the first service definition value object for the query

---

firstJson

```
public String firstJson()
```

Execute the query and return the first service definition JSON value.

Returns:

- execute the query and return the first service definition JSON value

TxnQuery

Package:

- com.avoka.tm.query

Class TxnQuery

- java.lang.Object
- com.avoka.tm.query.TxnQuery

```
public class TxnQuery
extends Object
```

Provides a transaction value object query class.

Examples

Please find the transaction query examples about list, get first and count below.

Transaction Query List Example

This Groovy example shows how to list transactions matching certain criteria ordered descending by id (maximum number of 150 records).

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

List<Txn> txns = new TxnQuery()
     .setFormVersionNumber("2.0")
     .setFetchLimit(150)
     .addOrderByDesc("id")
     .listValues()

// In JSON format
String txnsReady = new TxnQuery()
     .setFormVersionNumber("2.0")
     .setDeliveryStatus("Ready")
     .setUserLoginName("john.smith@avoka.com")
     .setSpaceName("Work Space")
     .listJson()
```

Transaction Query First Example

This Groovy example shows how to get first transaction matching certain criteria ordered descending by id.

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

Txn txn = new TxnQuery()
     .setFormVersionNumber("3.2")
     .addOrderByDesc("id")
     .firstValue()

// In JSON format
String txnReady = new TxnQuery()
     .setFormVersionNumber("2.0")
     .setDeliveryStatus(Txn.DELIVERY_READY)
     .firstJson()
```

Transaction Query Count Example

This Groovy example shows how to count all transactions matching certain criteria.

```
import com.avoka.tm.query.*

int readyCount = new TxnQuery()
```

```
     .setFormVersionNumber("2.0")
     .setDeliveryStatus(Txn.DELIVERY_READY)
     .count()
```

Transaction Query Job Example

The example below returns the list of transactions associated the job reference number.

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

List<Txn> txns = new TxnQuery()
     .setJobRefNumber("L9MKYT")
     .listValues()
```

The example below returns the list of transactions associated the specified job step.

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

Job job = new JobQuery()
     .setJobRefNumber("L9MKYT")
     .firstValue()

JobStep jobStep = job.jobSteps.get(1)

List<Txn> txns = new TxnQuery()
     .setJobStep(jobStep)
     .listValues()
```

Transaction Query "With" Example

For performance reasons TxnQuery does not load all of the Txn properties (formDataMap, formXml, receiptPdf, fileAttachList etc. or all of them) by default. Note the capability of `withXYZ()` methods for loading necessary transaction attributes.

This Groovy example shows how to query transactions and load their form data map and receipt pdf.

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

Txn txn = new TxnQuery()
     .setFormVersionNumber("2.0")
     .setDeliveryStatus(Txn.DELIVERY_READY)
     .withFormDataMap()
     .withReceiptPdf()
     .firstValue()

byte[] receiptPdf = txn.receiptPdf
```

This Groovy example shows how to query transactions and load all transaction attributes (fileAttachList, formDataMap, formXml, groupNames, propertyMap, receiptPdf).

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

String txnJson = new TxnQuery()
     .setFormVersionNumber("2.0")
     .setDeliveryStatus(Txn.DELIVERY_READY)
     .withAll()
     .firstJson()

String formXml = txn.formXml
byte[] receiptPdf = txn.receiptPdf
```

Since:

- 5.0.0

See Also:

- Txn

Constructor Summary

Constructors

Constructor and Description
TxnQuery()

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
TxnQuery	addOrderByAsc(String orderProperty) Add the sort order by ascending to the query.
TxnQuery	addOrderByDesc(String orderProperty) Add the sort order by descending to the query.
int	count() Execute a select count query and return the total number of records selected by the query.
String	firstJson() Execute the query and return the first transaction JSON value.
Txn	firstValue() Execute the query and return the first transaction value object for the query.
TxnQuery	hasProperty(String name) Set property name parameter.
String	listJson() Execute the query and return an transactions JSON array list.
List<Txn>	listValues() Execute the transaction query and return a list of Txn value objects.
TxnQuery	setAttachmentsStatus(String attachmentsStatus) Set the transaction attachment status query parameter [Required | Optional | Completed].
TxnQuery	setDeliveryStatus(String deliveryStatus) Set the transaction delivery status query parameter [Not Ready | Ready | Sent_Email | In Progress | Pending | Completed | Error | Undeliverable | Not Required].
TxnQuery	setEmailAddress(String emailAddress) Set the contact email address query parameters.
TxnQuery	setFetchLimit(int fetchLimit) Set the query fetch limit to limit the maximum number of records returned.
TxnQuery	setFormCode(String formCode) Set the form code query parameter.
TxnQuery	setFormStatus(String formStatus) Set the transaction form status query parameter [Assigned | Opened | Saved | Submitted | Completed | Expired | Abandoned].
TxnQuery	setFormVersionNumber(String formVersionNumber) Set the form version number query parameter.
TxnQuery	setId(Number id) Set the transaction id (PK) query parameter.
TxnQuery	setJobRefNumber(String jobRefNumber) Set the job reference number query parameter.
TxnQuery	setJobStep(JobStep jobStep) Set the job step query parameter.
TxnQuery	setPaymentStatus(String paymentStatus) Set the transaction payment status query parameter [Required | Completed | Error | Pending].

TxnQuery	setReceiptNumber(String receiptNumber) Set the transaction receipt number query parameter.
TxnQuery	setSpaceName(String spaceName) Set the form space name query parameter.
TxnQuery	setSubmitKey(String submitKey) Set the transaction submission key (GUID) query parameter.
TxnQuery	setTrackingCode(String trackingCode) Set the transaction tracking code query parameter.
TxnQuery	setTxn(Txn txn) Set the transaction query parameter.
TxnQuery	setTxnReferenceNumber(String txnReferenceNumber) Set the transaction reference number query parameter.
TxnQuery	setUserLoginName(String userLoginName) Set the transaction user's login name (username).
TxnQuery	setUserSaved(boolean userSaved) Set the user saved query parameter.
TxnQuery	withAll() Set the query to return the transaction with all the associated attachments map, form data map, form XML, group names, property map and receipt PDF data.
TxnQuery	withFileAttachList() Set the query to return the transaction with the associated file attachment list.
TxnQuery	withFormDataMap() Set the query to return the transaction with the associated form data map information.
TxnQuery	withFormXml() Set the query to return the transaction with the associated form XML information.
TxnQuery	withGroupNames() Set the query to return the transaction with the associated group names information.
TxnQuery	withPropertyMap() Set the query to return the transaction with the associated property map information.
TxnQuery	withReceiptPdf() Set the query to return the transaction with the associated receipt PDF data.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

TxnQuery

```
public TxnQuery()
```

Method Detail

setAttachmentsStatus

```
public TxnQuery setAttachmentsStatus(String attachmentsStatus)
```

Set the transaction attachment status query parameter [Required | Optional | Completed].

Parameters:

- `attachmentsStatus` - the attachments status

Returns:

- the transaction query

---

setEmailAddress

```
public TxnQuery setEmailAddress(String emailAddress)
```

Set the contact email address query parameters.

Parameters:

- `emailAddress` - the contact address query parameter

Returns:

- the transaction query

---

setDeliveryStatus

`public TxnQuery setDeliveryStatus(String deliveryStatus)`

Set the transaction delivery status query parameter [Not Ready | Ready | Sent_Email | In Progress | Pending | Completed | Error | Undeliverable | Not Required].

Parameters:

- `deliveryStatus` - the delivery status query parameter

Returns:

- the transaction query

---

setFormCode

`public TxnQuery setFormCode(String formCode)`

Set the form code query parameter.

Parameters:

- `formCode` - the form code query parameter

Returns:

- the transaction query

---

setFormStatus

`public TxnQuery setFormStatus(String formStatus)`

Set the transaction form status query parameter [Assigned | Opened | Saved | Submitted | Completed | Expired | Abandoned].

Parameters:

- `formStatus` - the form status query parameter

Returns:

- the transaction query

---

setId

`public TxnQuery setId(Number id)`

Set the transaction id (PK) query parameter.

Parameters:

- `id` - the transaction id (PK) query parameter

Returns:

- the transaction query

---

setTxn

`public TxnQuery setTxn(Txn txn)`

Set the transaction query parameter.

Parameters:

- `txn` - the transaction query parameter (required)

Returns:

- the transaction query

---

setJobStep

`public TxnQuery setJobStep(JobStep jobStep)`

Set the job step query parameter.

Parameters:

- `jobStep` - the job step query parameter

Returns:

- the transaction query

---

setJobRefNumber

`public TxnQuery setJobRefNumber(String jobRefNumber)`

Set the job reference number query parameter.

Parameters:

- `jobRefNumber` - the job reference number query parameter

Returns:

- the transaction query

---

setUserLoginName

`public TxnQuery setUserLoginName(String userLoginName)`

Set the transaction user's login name (username).

Parameters:

- `userLoginName` - the transaction user's login name (username).

Returns:

- the transaction query

---

setPaymentStatus

`public TxnQuery setPaymentStatus(String paymentStatus)`

Set the transaction payment status query parameter [Required | Completed | Error | Pending].

Parameters:

- `paymentStatus` - the transaction payment status query parameter

Returns:

- the transaction query

---

setReceiptNumber

`public TxnQuery setReceiptNumber(String receiptNumber)`

Set the transaction receipt number query parameter.

Parameters:

- `receiptNumber` - the transaction receipt number query parameter

Returns:

- the transaction query

---

setSubmitKey

`public TxnQuery setSubmitKey(String submitKey)`

Set the transaction submission key (GUID) query parameter.

Parameters:

- `submitKey` - the transaction submission key (GUID) query parameter

Returns:

- the transaction query

---

setTrackingCode

```
public TxnQuery setTrackingCode(String trackingCode)
```

Set the transaction tracking code query parameter.

Parameters:

- `trackingCode` - the transaction tracking code query parameter

Returns:

- the transaction query

---

setTxnReferenceNumber

```
public TxnQuery setTxnReferenceNumber(String txnReferenceNumber)
```

Set the transaction reference number query parameter.

Parameters:

- `txnReferenceNumber` - the transaction reference number query parameter

Returns:

- the transaction query

---

setFormVersionNumber

```
public TxnQuery setFormVersionNumber(String formVersionNumber)
```

Set the form version number query parameter.

Parameters:

- `formVersionNumber` - the form version number query parameter

Returns:

- the transaction query

---

setSpaceName

```
public TxnQuery setSpaceName(String spaceName)
```

Set the form space name query parameter.

Parameters:

- `spaceName` - the form space name query parameter

Returns:

- the transaction query

---

setUserSaved

```
public TxnQuery setUserSaved(boolean userSaved)
```

Set the user saved query parameter.

Parameters:

- `userSaved` - the the user saved query parameter

Returns:

- the transaction query

Since:

- 5.1.3

---

hasProperty

```
public TxnQuery hasProperty(String name)
```

Set property name parameter.

Parameters:

- `name` - the property name query parameter

Returns:

- the transaction query

Since:

- 5.0.1

---

setFetchLimit

```
public TxnQuery setFetchLimit(int fetchLimit)
```

Set the query fetch limit to limit the maximum number of records returned. The default query fetch limit is 100 records.

Parameters:

- `fetchLimit` - the query fetch limit

Returns:

- the transaction query

---

withAll

```
public TxnQuery withAll()
```

Set the query to return the transaction with all the associated attachments map, form data map, form XML, group names, property map and receipt PDF data.

Returns:

- the transaction query

---

withFormDataMap

```
public TxnQuery withFormDataMap()
```

Set the query to return the transaction with the associated form data map information.

Returns:

- the transaction query

---

withFormXml

```
public TxnQuery withFormXml()
```

Set the query to return the transaction with the associated form XML information.

Returns:

- the transaction query

---

withGroupNames

```
public TxnQuery withGroupNames()
```

Set the query to return the transaction with the associated group names information.

Returns:

- the transaction query

---

withPropertyMap

`public TxnQuery withPropertyMap()`

Set the query to return the transaction with the associated property map information.

Returns:

- the transaction query

---

withReceiptPdf

`public TxnQuery withReceiptPdf()`

Set the query to return the transaction with the associated receipt PDF data.

Returns:

- the transaction query

---

withFileAttachList

`public TxnQuery withFileAttachList()`

Set the query to return the transaction with the associated file attachment list.

Returns:

- the transaction query

---

addOrderByAsc

`public TxnQuery addOrderByAsc(String orderProperty)`

Add the sort order by ascending to the query.

Parameters:

- `orderProperty` - the property to sort by (required)

Returns:

- the transaction query

---

addOrderByDesc

`public TxnQuery addOrderByDesc(String orderProperty)`

Add the sort order by descending to the query.

Parameters:

- `orderProperty` - the property to sort by (required)

Returns:

- the transaction query

---

count

`public int count()`

Execute a select count query and return the total number of records selected by the query.

Returns:

- the total number of records selected by the query

---

listValues

`public List<Txn> listValues()`

Execute the transaction query and return a list of Txn value objects.

Returns:

- execute the transaction query and return a list of Txn value objects

---

listJson

```
public String listJson()
```

Execute the query and return an transactions JSON array list.

Returns:

- execute the query and return an transactions JSON array list

---

firstValue

```
public Txn firstValue()
```

Execute the query and return the first transaction value object for the query.

Returns:

- execute the query and return the first transaction value object for the query

---

firstJson

```
public String firstJson()
```

Execute the query and return the first transaction JSON value.

Returns:

- execute the query and return the first transaction JSON value

UserQuery

Package:

- com.avoka.tm.query

Class UserQuery

- java.lang.Object
- com.avoka.tm.query.UserQuery

```
public class UserQuery
extends Object
```

Provides a user value object query class.

Examples

Please find the user query examples about list, get first and count below.

User Query List Example

This Groovy example shows how to list active users sorted by first name and last name, with the result set limited to a maximum of 50 records.

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

List<User> users = new UserQuery()
    .setAccountStatus(User.STATUS_ACTIVE)
    .addOrderByAsc("firstName")
    .addOrderByAsc("lastName")
    .setFetchLimit(50)
    .listValues()

// In JSON format
String usersJson = new UserQuery()
    .setAccountStatus(User.STATUS_ACTIVE)
    .addOrderByAsc("firstName")
    .addOrderByAsc("lastName")
    .setFetchLimit(50)
    .listJson()
```

User Query First Example

This Groovy example shows how to get first user matching certain criteria.

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

User userJohn = new UserQuery()
    .setFirstName("John")
    .firstValue()
```

User Query Count Example

This Groovy example shows how to count all users which have an inactive status.

```
import com.avoka.tm.query.*

int inactiveCount = new UserQuery()
    .setAccountStatus(User.STATUS_INACTIVE)
    .count()
```

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`UserQuery()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`UserQuery`	`addOrderByAsc(String orderProperty)` Add the sort order by ascending to the query.
`UserQuery`	`addOrderByDesc(String orderProperty)` Add the sort order by descending to the query.
`int`	`count()` Execute a select count query and return the total number of records selected by the query.
`String`	`firstJson()` Execute the query and return the first user JSON value.
`User`	`firstValue()` Execute the query and return the first user value object for the query.
`String`	`listJson()` Execute the query and return an users JSON array list.
`List<User>`	`listValues()` Execute the user query and return a list of User value objects.
`UserQuery`	`setAccountStatus(String accountStatus)` Set the user account status [Active | Inactive | Locked | Locked Temporarily | Pending | Rejected] query parameter.
`UserQuery`	`setClientCode(String clientCode)` Set the user organization client code query parameter.
`UserQuery`	`setEmail(String email)` Set the user email query parameter.
`UserQuery`	`setFetchLimit(int fetchLimit)` Set the query fetch limit to limit the maximum number of records returned.
`UserQuery`	`setFirstName(String firstName)` Set the user first name query parameter.
`UserQuery`	`setGlobalAccess(boolean hasGlobalAccess)` Set the user has administrative global organization access.
`UserQuery`	`setId(Long id)` Set the user id (PK) query parameter.
`UserQuery`	`setLastName(String lastName)` Set the user last name query parameter.
`UserQuery`	`setLoginName(String loginName)` Set the user login name query parameter.
`UserQuery`	`setMobile(String mobile)` Set the user mobile number query parameter.
`UserQuery`	`setUserKey(String userKey)` Set the user key query parameter.
`UserQuery`	`setUserType(String userType)` Set the user type [Local | LDAP | SSO] query parameter.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

UserQuery

```
public UserQuery()
```

Method Detail

setId

```
public UserQuery setId(Long id)
```

Set the user id (PK) query parameter.

Parameters:

- `id` - the user id (PK) query parameter

Returns:

- the user query

---

setLoginName

```
public UserQuery setLoginName(String loginName)
```

Set the user login name query parameter.

Parameters:

- `loginName` - the user login name query parameter

Returns:

- the user query

---

setAccountStatus

```
public UserQuery setAccountStatus(String accountStatus)
```

Set the user account status [Active | Inactive | Locked | Locked Temporarily | Pending | Rejected] query parameter.

Parameters:

- `accountStatus` - the user account status [Active | Inactive | Locked | Locked Temporarily | Pending | Rejected] query parameter

Returns:

- the user query

---

setEmail

```
public UserQuery setEmail(String email)
```

Set the user email query parameter.

Parameters:

- `email` - the user email query parameter

Returns:

- the user query

---

setFirstName

```
public UserQuery setFirstName(String firstName)
```

Set the user first name query parameter.

Parameters:

- `firstName` - the user first name query parameter

Returns:

- the user query

setLastName

`public UserQuery setLastName(String lastName)`

Set the user last name query parameter.

Parameters:

- `lastName` - the user last name query parameter

Returns:

- the user query

---

setMobile

`public UserQuery setMobile(String mobile)`

Set the user mobile number query parameter.

Parameters:

- `mobile` - the user mobile number query parameter

Returns:

- the user query

---

setClientCode

`public UserQuery setClientCode(String clientCode)`

Set the user organization client code query parameter.

Parameters:

- `clientCode` - the user client organization code query parameter

Returns:

- the user query

---

setUserKey

`public UserQuery setUserKey(String userKey)`

Set the user key query parameter.

Parameters:

- `userKey` - the user key query parameter

Returns:

- the user query

---

setUserType

`public UserQuery setUserType(String userType)`

Set the user type [Local | LDAP | SSO] query parameter.

Parameters:

- `userType` - the user type [Local | LDAP | SSO] query parameter

Returns:

- the user query

---

setGlobalAccess

`public UserQuery setGlobalAccess(boolean hasGlobalAccess)`

Set the user has administrative global organization access.

Parameters:

- `hasGlobalAccess` - the user has administrative global organization access

Returns:

- the user query

---

setFetchLimit

```
public UserQuery setFetchLimit(int fetchLimit)
```

Set the query fetch limit to limit the maximum number of records returned. The default query fetch limit is 100 records.

Parameters:

- `fetchLimit` - the query fetch limit

Returns:

- the user query

---

addOrderByAsc

```
public UserQuery addOrderByAsc(String orderProperty)
```

Add the sort order by ascending to the query.

Parameters:

- `orderProperty` - the property to sort by (required)

Returns:

- the user query

---

addOrderByDesc

```
public UserQuery addOrderByDesc(String orderProperty)
```

Add the sort order by descending to the query.

Parameters:

- `orderProperty` - the property to sort by (required)

Returns:

- the user query

---

count

```
public int count()
```

Execute a select count query and return the total number of records selected by the query.

Returns:

- the total number of records selected by the query

---

listValues

```
public List<User> listValues()
```

Execute the user query and return a list of User value objects.

Returns:

- execute the user query and return a list of User value objects

---

listJson

```
public String listJson()
```

Execute the query and return an users JSON array list.

Returns:

- execute the query and return an users JSON array list

---

firstValue

```
public User firstValue()
```

Execute the query and return the first user value object for the query.

Returns:

- execute the query and return the first user value object for the query

---

firstJson

```
public String firstJson()
```

Execute the query and return the first user JSON value.

Returns:

- execute the query and return the first user JSON value

com.avoka.tm.security

Classes in Package com.avoka.tm.security

- Saml2Parser
- Saml2ParserResult
- SsoAuthToken

Saml2Parser

Package:

- com.avoka.tm.security

Class Saml2Parser

- java.lang.Object
- com.avoka.tm.security.Saml2Parser

```
public class Saml2Parser
extends Object
```

Provides a SAML2 (Security Assertion Markup Language) response parser - a helper class for SSO Filter / Security Manager **Get SSO Auth Token** script.

A SAML2 Response is sent by an Identity Provider (like Microsoft ADFS server) to a Service Provider (TM). The Identity Provider http POST as SAML Response to TM (Service Provider) receives the SAML token as HttpServletRequest request parameter. The parser first Base64 decodes the SAML response converting the raw response to XML. Note the response structure varies greatly from different types of Identity Providers software and the way they are configured.

Saml2Parser Usage

Example 1 and 2 demostrate the use of the parser.

There is a static convenience method has SAMLResponse(request). Requests that don't have a SAMLResponse may be redirected back to the Identity Provider.

A parser is constructed using **new Saml2Parser()** then chains Fluent style properties setters. These properties include setting the validation certificate and the keystore holding the private encryption key. In a live TM system these are configured in the associated Security Manager's Certificates tab.

Processing checks are run to validate the SAML response. By default the class will try to run **all** the checks. If your token has Assertion Signatures but no Response Signature then use skip skipResonseSignatureValidation(), see example 2 below. It is possible for unit testing to save a SAMLResponse captured in Chrome Tools or this tool. The issue with using this response to run a test in the future it will fail because of the timestamp checks. To get it working will have to skip a number the date checks listed in the table below.

Once you have completed calling the setters and skip method you can call **parse(request)**. This validates the class has been setup correctly and if configured incorrectly will throw an **IllegalArgumentException**. Otherwise it will try validating and parsing the SAMLResponse, returning a **Saml2ParserResult** which if successful **result.isValid** will contain an **result.ssoAuthToken**. The SsoAuthToken holds the user attributes (name, email, groups etc) parsed from the SAML Token. It is returned by the Security Manager **Get SSO Auth Token** script.

Logging

The **Saml2ParserResult** provides the following properties which are useful for troubleshooting:

responseRaw: The raw SAMLResponse (String) taken from the request parameter. This can be useful for debug the Saml2Parser in a separate unit test. **responseBase64**: The SAMLResponse XML (String) after it has been Base64 decoded. This is useful to inspect the **error**: This will be present when result.isValid is false, it will be null when result.isValid is true. It holds the Throwable error that occurred during validation and parsing. The error can be inspected to the cause of failure. The SecurityLogger can log this error and display the failure message and stack trace. **infoLog**: A summary level log that records when higher level sections have been started or completed. Use the Security Logger to store this log. Below is a Sample info log.

```
 Start parse()
 Start validateAndParse()
 Skipping ResponseSignatureValidation
 pre: responseSucessfulValidation(response
 Skipping responseDateCheck
 Assertion signature validated
 Skipping AssertionConditionCheck
 Skipping assertionAuthnStatementCheck
 Skipping assertionSubjectCheck
 Execution Duration(ms): 16
```

debugLog: A detailed level log in text format, includes all the info log plus a lot of logging in the validation and parsing. Below shows some of the extra detail captured in the detail log.

```
 New AttributeStatement
 parseAttributeStatements(): unencrypted attribute.getName() http://schemas.xmlsoap.org/ws/2005/05/identity/claims
/givenname
 parseAttributeStatements(): unencrypted attribute.getName() http://schemas.xmlsoap.org/ws/2005/05/identity/claims
/upn
 parseAttributeStatements(): unencrypted attribute.getName() http://schemas.xmlsoap.org/ws/2005/05/identity/claims
/surname
 parseAttributeStatements(): unencrypted attribute.getName() http://schemas.xmlsoap.org/ws/2005/05/identity/claims
/emailaddress
 parseAttributeStatements(): unencrypted attribute.getName() http://schemas.xmlsoap.org/claims/Group
 parseAttributeStatements(): unencrypted attribute.getName() http://schemas.microsoft.com/ws/2008/06/identity/claims
/windowsaccountname
 unencrypted attributes added for AttributeStatement
 Finished adding encrypted attributes for AttributeStatement
 Start parseAttributes(
```

```
 getSSOAuthToken() XMLObject getFirstChild().getTextContent(): peter
 getSSOAuthToken() XMLObject getFirstChild().getTextContent(): peter.panda@test.avoka.com
 getSSOAuthToken() XMLObject getFirstChild().getTextContent(): panda
 getSSOAuthToken() XMLObject getFirstChild().getTextContent(): peter.panda@test.avoka.com
 getSSOAuthToken() XMLObject getFirstChild().getTextContent(): CN=TransactionManagerAccess,OU=Group,OU=Marketing,
DC=test,DC=avoka,DC=com
 getSSOAuthToken() XMLObject getFirstChild().getTextContent(): peter.panda
 parseAssertions(): subject not encryptedparseAssertions(): NameID=org.opensaml.saml.saml2.core.impl.
NameIDImpl@2e11485, NameID.getValue=:peter.panda
 Execution Duration(ms): 16
```

By default the Saml2Parser does **not** do any logging direct to the TM or server logs. Use the SecurityLogger to capture these properties. You must explicitly add the output to the SecurityLogger (logger.info result.responseRaw) as per Example 1 below.

Example 1

This is called by the Security Manager **Get SSO Auth Token** script

```
import com.avoka.core.groovy.SecurityLogger as logger

import com.avoka.tm.security.*
import com.avoka.fc.core.util.RedirectUtils
import com.avoka.fc.core.util.PortalUtils
import com.avoka.fc.core.entity.SecurityManager
import com.avoka.fc.core.util.RedirectUtils

// Stores the Entry URL into the session which is used by the Auth Ok Response Script.
if (!Saml2Parser.hasSamlToken(request)) {
    logger.info "No SAML Token, Storing sessionEntryUrl and redirecting to ADFS server"
    RedirectUtils.storeSessionEntryUrl(request)
    return null
}

Saml2ParserResult result = new Saml2Parser()
                        .setValidationCertData(securityManager.getSsoValidatorCertData())
                        .setKeystoreData(securityManager.getSsoKeystoreData())
                        .setKeystorePassword(securityManager.getSsoKeystorePassword())
                        .setPrivateKeyAlias(securityManager.getSsoPrivateKeyAlias())
                        .setPrivateKeyPassword(securityManager.getSsoPrivateKeyPassword())
                        .setGroupAttribName("http://schemas.xmlsoap.org/claims/Group")
                        .setRPIdentifier("https://{TM_Server_IP}/{PORTAL_Context}")
                        .parse(request)

if (result.isValid) {
    return result.ssoAuthToken

} else {
     logger.debug result.debugLog
     logger.debug result.responseRaw
     logger.info result.validationErrors.join("\n")

     logger.info "Redirecting to " + PortalUtils.getNotAuthorizedPath(portal)
     throw new RedirectException(PortalUtils.getNotAuthorizedPath(portal))
}
```

Example 2

This Groovy example shows how to bypass a some of the inbuilt checking when parsing the SAMLToken

```
Saml2ParserResult result = new Saml2Parser()
                        .setValidationCertData(securityManager.getSsoValidatorCertData())
                        .setKeystoreData(securityManager.getSsoKeystoreData())
                        .setKeystorePassword(securityManager.getSsoKeystorePassword())
                        .setPrivateKeyAlias(securityManager.getSsoPrivateKeyAlias())
                        .setPrivateKeyPassword(securityManager.getSsoPrivateKeyPassword())
                        .setGroupAttribName("http://schemas.xmlsoap.org/claims/Group")
                        .setRPIdentifier("https://{TM_Server_IP}/{PORTAL_Context}")
                        .setUrlDecodeResponse()
                        .skipRequireEncryption()
                        .skipResponseDateCheck()
                        .skipAssertionSubjectCheck()
                        .skipAssertionAuthnStatementCheck()
                        .skipAssertionConditionCheck()
                        .skipAssertionSignatureValidation()
```

```
                        .parse(request)
```

SAML Token Structure

Example SAML2 Tokens can be found on www.samltool.com.

The following table show is the XML Node Hierarchy of the SAML Response

```
 Node Heirachy                                  Description                                          Skip Method

-------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------
 Response                                       Attr: IssueInstant tstamp
skipResponseDateCheck()
                                                      Destination
skipDestinationEndpointCheck()

     Signature (optional):                      Signature for the whole response.
skipResonseSignatureValidation()

     Issuer                                     Issuing server url eg ADFS server

     Assertion (List):                          Holder for the user details.
                                                Normally Encrypted.
skipRequireEncryption()

         Signature (optional):                  The signature for the assertion.
skipAssertionSignatureValidation()

         Issuer:                                url of the issuing server

         Subject:
skipAssertionSubjectCheck()
             NameID                             This gets written to ssoAuthToken.username
             SubjectConfirmation
                 SubjectConfirmationData        Attr:  notOnOrAfter (tstamp)
skipAssertionSubjectCheck()
                                                recipient should be the same as RPIdenifier
skipAssertionSubjectCheck() or skipDestinationEndpointCheck()

         Conditions:                            Attr:  notBefore and notOnOrAfter (tstamp)
skipAssertionConditionCheck()
             AudienceRestriction
                 Audience (List):               Check Relying Party Identity in List
skipAssertionConditionCheck() also requires parser.setRPIdentifier()

         AttributeStatement
             Attribute (List):                  Contains the details for each user attrib.

         AuthnStatement                         Attr: NotOnOrAfter tstamp
skipAssertionAuthnStatementCheck()
                                                      IssueInstant
```

Since:

- 5.1.4

Constructor Summary

Constructors

Constructor and Description
`Saml2Parser()`

Method Summary

All Methods Static Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`static boolean`	`hasSamlToken(HttpServletRequest request)` Returns true if the request has a SAML2 SAMLRequest or SAMLResponse request parameters.
`Saml2ParserResult`	`parse(HttpServletRequest request)` Parse the SAMLResponse from the given POST request.
`Saml2Parser`	`setDestinationEndpoint(String destinationEndpoint)` Sets the DesinationEndpoint for Response **Destination** attribute and Assertion Subject.
`Saml2Parser`	`setGroupAttribName(String groupAttribName)`

	Sets the Group Attribute Name.
Saml2Parser	setKeystoreData(byte[] keystoreData) Sets the keystore that holds the private key and public Certificate keys.
Saml2Parser	setKeystorePassword(String keystorePassword) Sets the keystore password.
Saml2Parser	setPrivateKeyAlias(String privateKeyAlias) Sets the Private Key Alias for the keystore.
Saml2Parser	setPrivateKeyPassword(String privateKeyPassword) Sets the Private Key Password.
Saml2Parser	setRPIdentifier(String rpIdentifier) Sets the Relying Party Identifier used by the security manager.
Saml2Parser	setUrlDecodeResponse() Specify whether to URL decode the SAML response befpre Base64 decoding.
Saml2Parser	setValidationCertData(byte[] validatorCertData) Sets the token signing certificate used for validating the SAML2 Signature.
Saml2Parser	skipAssertionAuthnStatementCheck() Specify whether to skip the execution of the Assertion AuthnStatment checks.
Saml2Parser	skipAssertionConditionCheck() Specify whether to skip the execution of checks associated with the Assertion Condition.
Saml2Parser	skipAssertionSignatureValidation() Specify whether to skip the execution of the Assertion Signature Validation checks.
Saml2Parser	skipAssertionSubjectCheck() Specify whether to skip the execution of checks associated with the Assertion Subject.
Saml2Parser	skipDestinationEndpointCheck() Specify whether to skip the execution of checks associated with the Response.
Saml2Parser	skipRequireEncryption() Specify whether to skip the encryption requirement and allow an unencypted SAML token to be parsed.
Saml2Parser	skipResponseDateCheck() Specify whether to skip the execution of the Response Date Check on the Response:@IssueInstant timestamp attribute.
Saml2Parser	skipResponseSignatureValidation() Specify whether to skip the execution of the Response Signature Validation.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Saml2Parser

```
public Saml2Parser()
```

Method Detail

hasSamlToken

```
public static boolean hasSamlToken(HttpServletRequest request)
```

Returns true if the request has a SAML2 SAMLRequest or SAMLResponse request parameters.

Parameters:

- `request` - the HttpServletRequest Containing the SAMLResponse (required)

Returns:

- true if has a SAMLRequest or SAMLResponse request parameter

setValidationCertData

`public Saml2Parser setValidationCertData(byte[] validatorCertData)`

Sets the token signing certificate used for validating the SAML2 Signature.

Parameters:

- `validatorCertData` - byte[] the certificate used for validating the SAML2 Signature (Optional)

Returns:

- the Saml2Parser

---

setKeystoreData

`public Saml2Parser setKeystoreData(byte[] keystoreData)`

Sets the keystore that holds the private key and public Certificate keys.

Parameters:

- `keystoreData` - byte[] that holds the private key and keys (Optional)

Returns:

- the Saml2Parser

---

setKeystorePassword

`public Saml2Parser setKeystorePassword(String keystorePassword)`

Sets the keystore password.

Parameters:

- `keystorePassword` - String

Returns:

- the Saml2Parser

---

setPrivateKeyAlias

`public Saml2Parser setPrivateKeyAlias(String privateKeyAlias)`

Sets the Private Key Alias for the keystore.

Parameters:

- `privateKeyAlias` - String

Returns:

- the Saml2Parser

---

setPrivateKeyPassword

`public Saml2Parser setPrivateKeyPassword(String privateKeyPassword)`

Sets the Private Key Password.

Parameters:

- `privateKeyPassword` - String

Returns:

- the Saml2Parser

---

setRPIdentifier

`public Saml2Parser setRPIdentifier(String rpIdentifier)`

Sets the Relying Party Identifier used by the security manager.

Parameters:

- `rpIdentifier` - (Optional) String for this space / environment and eg https://tm.server.com/sso

Returns:

- the Saml2Parser

---

setGroupAttribName

```
public Saml2Parser setGroupAttribName(String groupAttribName)
```

Sets the Group Attribute Name.
If the SAML2 attribute parser has groupAttributeName configured then the `SSOAuthenticationToken` will any parsed groups as Spring `GrantedAuthoritys`.

Parameters:

- `groupAttribName` - String

Returns:

- the Saml2Parser

---

setDestinationEndpoint

```
public Saml2Parser setDestinationEndpoint(String destinationEndpoint)
```

Sets the DesinationEndpoint for Response **Destination** attribute and Assertion Subject. SubjectConfirmationData **recipient** attribute.

Used for unit testing, This value that is normally used is request.getRequestURL()

Parameters:

- `destinationEndpoint` - the destinationEndpoint to set.

Returns:

- the Saml2Parser

---

setUrlDecodeResponse

```
public Saml2Parser setUrlDecodeResponse()
```

Specify whether to URL decode the SAML response befpre Base64 decoding.

Returns:

- the Saml2Parser

---

skipRequireEncryption

```
public Saml2Parser skipRequireEncryption()
```

Specify whether to skip the encryption requirement and allow an unencypted SAML token to be parsed.

Returns:

- the Saml2Parser

---

skipResponseSignatureValidation

```
public Saml2Parser skipResponseSignatureValidation()
```

Specify whether to skip the execution of the Response Signature Validation.

Returns:

- the Saml2Parser

---

skipResponseDateCheck

```
public Saml2Parser skipResponseDateCheck()
```

Specify whether to skip the execution of the Response Date Check on the Response:@IssueInstant timestamp attribute.

Returns:

- the Saml2Parser

---

skipAssertionSignatureValidation

`public Saml2Parser skipAssertionSignatureValidation()`

Specify whether to skip the execution of the Assertion Signature Validation checks.

Returns:

- the Saml2Parser

---

skipAssertionAuthnStatementCheck

`public Saml2Parser skipAssertionAuthnStatementCheck()`

Specify whether to skip the execution of the Assertion AuthnStatment checks.

Returns:

- the Saml2Parser

---

skipAssertionConditionCheck

`public Saml2Parser skipAssertionConditionCheck()`

Specify whether to skip the execution of checks associated with the Assertion Condition.

Returns:

- the Saml2Parser

---

skipAssertionSubjectCheck

`public Saml2Parser skipAssertionSubjectCheck()`

Specify whether to skip the execution of checks associated with the Assertion Subject.

Returns:

- the Saml2Parser

---

skipDestinationEndpointCheck

`public Saml2Parser skipDestinationEndpointCheck()`

Specify whether to skip the execution of checks associated with the Response. **Destination** attribute and the Assertion SubjectConfirmationData **recipient**.

Returns:

- the Saml2Parser

---

parse

`public Saml2ParserResult parse(HttpServletRequest request)`

Parse the SAMLResponse from the given POST request.

Parameters:

- `request` - the servlet request which contains the SAML Token (required)

Returns:

- the Saml2ParserResult

Saml2ParserResult

Package:

- com.avoka.tm.security

Class Saml2ParserResult

- java.lang.Object
- com.avoka.tm.security.Saml2ParserResult

```
public class Saml2ParserResult
extends Object
```

The Saml2ParserResult is returned from the Saml2Paser.parse(request) method.

Since:

- 5.1.4

See Also:

- Saml2Parser

Field Summary

Fields

Modifier and Type	Field and Description
String	debugLog contains a detailed log, including the infoLog entries of the parse operation.
Throwable	error the Throwable when an error occurs during the parsing.
String	infoLog contains a summary log of the parse operation.
boolean	isValid The isValid is true when the parsing has completed successfully without error.
String	responseB64Decoded The SAMLResponse XML (String) after it has been Base64 decoded.
String	responseRaw The raw SAMLResponse (String) taken from the request parameter.
SsoAuthToken	ssoAuthToken The ssoAuthToken (SsoAuthToken) holds the parsed user details from the SAML Response.
List<String>	validationErrors contains a List of the validation error messages.

Constructor Summary

Constructors

Constructor and Description
Saml2ParserResult(SsoAuthToken ssoAuthToken, String responseRaw, String responseB64Decoded, Throwable error, List<String> validationErrors, String debugLog, String infoLog) Creates a immutable Saml2ParserResult.

Method Summary

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

---

isValid

```
public final boolean isValid
```

The isValid is true when the parsing has completed successfully without error.

---

ssoAuthToken

```
public final SsoAuthToken ssoAuthToken
```

The ssoAuthToken (SsoAuthToken) holds the parsed user details from the SAML Response. If isValid == false then it will be null.
If is valid the Security Manager **Get SSO Auth Token** should return this object

---

responseRaw

```
public final String responseRaw
```

The raw SAMLResponse (String) taken from the request parameter.
Save this to the SecurityLogger when troubleshooting. This can be useful for debug the Saml2Parser in a separate unit test

---

responseB64Decoded

```
public final String responseB64Decoded
```

The SAMLResponse XML (String) after it has been Base64 decoded. Save this to the SecurityLogger it is useful for troubleshooting the structure of the SAML token.

---

error

```
public final Throwable error
```

the Throwable when an error occurs during the parsing.

---

debugLog

```
public final String debugLog
```

contains a detailed log, including the infoLog entries of the parse operation.

---

infoLog

```
public final String infoLog
```

contains a summary log of the parse operation.

---

validationErrors

```
public final List<String> validationErrors
```

contains a List of the validation error messages.

Constructor Detail

Saml2ParserResult

```
public Saml2ParserResult(SsoAuthToken ssoAuthToken,
                         String responseRaw,
                         String responseB64Decoded,
                         Throwable error,
                         List<String> validationErrors,
                         String debugLog,
                         String infoLog)
```

Creates a immutable Saml2ParserResult.

Parameters:

- `ssoAuthToken` - the SsoAuthToken that holds the parsed user details from the SAML Response.
- `responseRaw` - the String raw SAMLResponse taken from the request parameter.
- `responseB64Decoded` - the String SAMLResonse XML after it is has been Base64 decoded.
- `error` - the Throwable associated with a failure to parse the SAML Response
- `validationErrors` - a String List or validationError entries. Each String entry contains a type, a path to the node in the SAML Response relates to, an error context message, if an exception was caught the e.message() and the stack trace.
- `debugLog` - the String detailed log including the infoLog entries for the parse operation
- `infoLog` - the String summary log of the parse operation.

SsoAuthToken

Package:

- com.avoka.tm.security

Class SsoAuthToken

- java.lang.Object
- com.avoka.tm.security.SsoAuthToken

```
public class SsoAuthToken
extends Object
```

Provides an SSO Authentication Token returned by the Saml2Parser in the Saml2ParserResult.

Note this URL context path pre-pending should be done by the calling code and will not be done by this class.

Since:

- 5.1.4

Constructor Summary

Constructors

Constructor and Description
`SsoAuthToken(String username)` Creates a new SSO Auth Token for the specified login name.
`SsoAuthToken(String username, Collection<GrantedAuthority> authorities)` Creates a new SSO Auth Token for the specified login name and granted authorities (groups).
`SsoAuthToken(String username, Map<String,Object> attributes)` Creates a new SSO Auth Token for the specified login name and attributes.
`SsoAuthToken(String username, Map<String,Object> attributes, List<String> groups)` Creates a new SSO Auth Token for the specified login name, attributes and groups the user belongs to.
`SsoAuthToken(String username, Map<String,Object> attributes, Set<String> groups)` Creates a new SSO Auth Token for the specified login name, attributes and groups the user belongs to.

Method Summary

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

SsoAuthToken

```
public SsoAuthToken(String username)
```

Creates a new SSO Auth Token for the specified login name.

Parameters:

- `username` - the login name

SsoAuthToken

```
public SsoAuthToken(String username,
                    Map<String,Object> attributes)
```

Creates a new SSO Auth Token for the specified login name and attributes.

Parameters:

- `username` - the login name
- `attributes` - the user attributes

SsoAuthToken

```
public SsoAuthToken(String username,
                    Collection<GrantedAuthority> authorities)
```

Creates a new SSO Auth Token for the specified login name and granted authorities (groups).

Parameters:

- `username` - the login name
- `authorities` - the uses set of granted authorities or groups.

SsoAuthToken

```
public SsoAuthToken(String username,
                    Map<String,Object> attributes,
                    Set<String> groups)
```

Creates a new SSO Auth Token for the specified login name, attributes and groups the user belongs to.

Parameters:

- `username` - the login name
- `attributes` - the user attributes
- `groups` - the set of groups the user belongs to, will be marshaled into granted authorities.

SsoAuthToken

```
public SsoAuthToken(String username,
                    Map<String,Object> attributes,
                    List<String> groups)
```

Creates a new SSO Auth Token for the specified login name, attributes and groups the user belongs to.

Parameters:

- `username` - the login name
- `attributes` - the user attributes
- `groups` - the set of groups the user belongs to, will be marshaled into granted authorities.

com.avoka.tm.svc

Classes in Package com.avoka.tm.svc

- DeliveryTxnBuilder
- Emailer
- ErrorLogger
- EventLogger
- GroovyServiceInvoker
- PropertyBuilder
- ReceiptSvc
- ServiceInvoker
- SvcConnUpdater
- TrackingCodeBuilder
- TxnBuilder
- TxnCheckpointSvc
- TxnUpdater
- UserBuilder

DeliveryTxnBuilder

Package:

- com.avoka.tm.svc

Class DeliveryTxnBuilder

- java.lang.Object
- com.avoka.tm.svc.DeliveryTxnBuilder

```
public class DeliveryTxnBuilder
extends Object
```

Provides a delivery transaction builder class.

Delivery only transactions are intended for delivering transaction data to systems of record. These transactions are not included in analytics and do not consume transaction licensing units. The are typically used in collaboration jobs where you have many assigned tasks being completed, but the results and associated file attachments are rolled up into one delivery transaction.

Examples

Provides a job delivery transaction example, where a transaction is created specifically for delivery purposes.

```
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

String deliveryXml = ...
FileAttach fileAttach = ...
JobAction jobAction = ...

Txn txn = new DeliveryTxnBuilder()
    .setFormCode("FTX-CCA")
    .setFormXml(deliveryXml)
    .setJobAction(jobAction)
    .addFileAttach(fileAttach)
    .build()
```

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`DeliveryTxnBuilder()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`DeliveryTxnBuilder`	`addFileAttach(FileAttach fileAttach)` Add the file attachment to the transaction (optional).
`DeliveryTxnBuilder`	`addFormDataExtract(String name, String value)` Add form data (data extract) to the transaction.
`Txn`	`build()` Build a delivery transaction with a Delivery Status of "Ready" using the specified parameters.
`DeliveryTxnBuilder`	`includeAllJobFileAttach()` Specify to included all the job transaction file attachments with the delivery transaction (optional).
`DeliveryTxnBuilder`	`setFormCode(String formCode)` Set the transaction form code (required).

DeliveryTxnBuilder	setFormXml(String formXml) Set the transaction form XML data to be delivered (optional).
DeliveryTxnBuilder	setJobAction(JobAction jobAction) Set the job action and job to be associated with the transaction (optional).
DeliveryTxnBuilder	setReceiptData(byte[] receiptData) Set the PDF receipt data to be delivered (optional).

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DeliveryTxnBuilder

```
public DeliveryTxnBuilder()
```

Method Detail

setFormCode

```
public DeliveryTxnBuilder setFormCode(String formCode)
```

Set the transaction form code (required).

Parameters:

- formCode - the formCode to set

Returns:

- delivery transaction builder

---

setFormXml

```
public DeliveryTxnBuilder setFormXml(String formXml)
```

Set the transaction form XML data to be delivered (optional). IF not specified a empty form XML document will be delivered.

Parameters:

- formXml - the form XML data

Returns:

- delivery transaction builder

---

setReceiptData

```
public DeliveryTxnBuilder setReceiptData(byte[] receiptData)
```

Set the PDF receipt data to be delivered (optional). If not specified a blank PDF receipt will be delivered.

Parameters:

- receiptData - the PDF receipt data

Returns:

- delivery transaction builder

---

setJobAction

```
public DeliveryTxnBuilder setJobAction(JobAction jobAction)
```

Set the job action and job to be associated with the transaction (optional).

Parameters:

- jobAction - the job action to associate the transaction with

Returns:

- delivery transaction builder

---

addFileAttach

```
public DeliveryTxnBuilder addFileAttach(FileAttach fileAttach)
```

Add the file attachment to the transaction (optional).

Parameters:

- `fileAttach` - the file attachment to add (required)

Returns:

- delivery transaction builder

---

includeAllJobFileAttach

```
public DeliveryTxnBuilder includeAllJobFileAttach()
```

Specify to included all the job transaction file attachments with the delivery transaction (optional).

Returns:

- delivery transaction builder

---

addFormDataExtract

```
public DeliveryTxnBuilder addFormDataExtract(String name,
                                             String value)
```

Add form data (data extract) to the transaction.

Parameters:

- `name` - form data (extract) name
- `value` - form data (extract) value

Returns:

- delivery transaction builder

---

build

```
public Txn build()
```

Build a delivery transaction with a Delivery Status of "Ready" using the specified parameters.

Returns:

- a delivery transaction with a Delivery Status of "Ready" using the specified parameters

Emailer

Package:

- com.avoka.tm.svc

Class Emailer

- java.lang.Object
- com.avoka.tm.svc.Emailer

```
public class Emailer
extends Object
```

Provides an Email service.

Examples

Please find the email service examples about sending and queuing below.

Send Email

This example shows how to send an email by setting the transaction id to resolve the email service with.

```
import com.avoka.tm.svc.Emailer

new Emailer()
   .setToAddress("john@doe.com")
   .setTxnId(3L)
   .setSubject("Subject")
   .setMessage("Hello, John Doe! How are you?")
   .sendEmail()
```

Queue Email

This example shows how to queue email.

```
import com.avoka.tm.svc.Emailer

new Emailer()
   .setToAddress("john@doe.com")
   .setReplyToAddress("reply@to.com")
   .setSubject("Subject")
   .setMessage("Hello, John Doe! How are you?")
   .setAttachmentMap(attachments)
   .queueEmail()
```

Send Saved Form Email

This example shows how to send email using the "Email Saved Form Message" and "Email Saved Form Subject" templates.

When using the Form Receipt Template you must specify the transaction so that the service can resolve the email template configurations.

```
import com.avoka.tm.svc.Emailer

new Emailer()
   .setTxn(txn)
   .setToAddress("john@doe.com")
   .setSavedFormTemplate(true)
   .sendEmail()
```

Send Form Receipt Email

This example shows how to queue email using the "Email Form Receipt Message" and "Email Form Receipt Subject" templates. Please note the PDF may be included in the email depending upon the associated transactions form configuration.

When using the Form Receipt Template you must specify the transaction so that the service can resolve the email template configurations.

```
import com.avoka.tm.svc.Emailer

new Emailer()
   .setTxn(txn)
   .setToAddress("john@doe.com")
   .setFormReceiptTemplate(true)
   .ququeEmail()
```

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`Emailer()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`void`	`queueEmail()` Add the email to the email queue.
`void`	`sendEmail()` Send the email immediately (synchronously).
`Emailer`	`setAttachmentMap(Map<String,byte[]> attachmentMap)` Set the email file attachments map, keyed on file name.
`Emailer`	`setBccAddress(String bccAddress)` Set the BCC send to email address.
`Emailer`	`setCcAddress(String ccAddress)` Set the CC send to email address.
`Emailer`	`setFormReceiptTemplate(boolean userFormReceiptTemplate)` Set to use the Email Form Receipt templates.
`Emailer`	`setFromAddress(String fromAddress)` Set the address to send the email from.
`Emailer`	`setMessage(String message)` Set the email message body.
`Emailer`	`setReplyToAddress(String replyToAddress)` Set the reply to email address.
`Emailer`	`setSavedFormTemplate(boolean useSavedFormTemplate)` Set to use Email Saved Form templates.
`Emailer`	`setSubject(String subject)` Set the email subject line.
`Emailer`	`setToAddress(String toAddress)` Set the address to send the email to.
`Emailer`	`setTxn(Txn txn)` Set the transaction to resolve the email service with.
`Emailer`	`setTxnId(Long txnId)` Set the transaction id to resolve the email service with.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

Emailer

```
public Emailer()
```

Method Detail

setSubject

```
public Emailer setSubject(String subject)
```

Set the email subject line.

Parameters:

- `subject` - the email subject

Returns:

- the email service

---

setMessage

```
public Emailer setMessage(String message)
```

Set the email message body.

Parameters:

- `message` - the email message body

Returns:

- the email service

---

setToAddress

```
public Emailer setToAddress(String toAddress)
```

Set the address to send the email to.

Parameters:

- `toAddress` - the address to send the email to

Returns:

- the email service

---

setBccAddress

```
public Emailer setBccAddress(String bccAddress)
```

Set the BCC send to email address.

Parameters:

- `bccAddress` - the BCC send to email address

Returns:

- the email service

---

setCcAddress

```
public Emailer setCcAddress(String ccAddress)
```

Set the CC send to email address.

Parameters:

- `ccAddress` - the CC send to email address

Returns:

- the email service

setFromAddress

`public Emailer setFromAddress(String fromAddress)`

Set the address to send the email from.

Parameters:

- `fromAddress` - the address to send the email from

Returns:

- the email service

---

setReplyToAddress

`public Emailer setReplyToAddress(String replyToAddress)`

Set the reply to email address.

Parameters:

- `replyToAddress` - the reply to email address

Returns:

- the email service

---

setAttachmentMap

`public Emailer setAttachmentMap(Map<String,byte[]> attachmentMap)`

Set the email file attachments map, keyed on file name.

Parameters:

- `attachmentMap` - the email file attachments map, keyed on file name

Returns:

- the email service

---

setTxn

`public Emailer setTxn(Txn txn)`

Set the transaction to resolve the email service with.

Parameters:

- `txn` - the transaction to resolve the email service with

Returns:

- the email service

---

setTxnId

`public Emailer setTxnId(Long txnId)`

Set the transaction id to resolve the email service with.

Parameters:

- `txnId` - the transaction id to resolve the email service with

Returns:

- the email service

---

setSavedFormTemplate

`public Emailer setSavedFormTemplate(boolean useSavedFormTemplate)`

Set to use Email Saved Form templates. Please note you must also specify the Txn when using this property.

Parameters:

- `useSavedFormTemplate` - Email Saved Form templates.

Returns:

- the email service

Since:

- 5.1.0

---

setFormReceiptTemplate

```
public Emailer setFormReceiptTemplate(boolean userFormReceiptTemplate)
```

Set to use the Email Form Receipt templates. Please note you must also specify the Txn when using this property.

Parameters:

- `userFormReceiptTemplate` - use Email Form Receipt templates

Returns:

- the email service

Since:

- 5.1.0

---

sendEmail

```
public void sendEmail()
```

Send the email immediately (synchronously).

---

queueEmail

```
public void queueEmail()
```

Add the email to the email queue.

ErrorLogger

Package:

- com.avoka.tm.svc

Class ErrorLogger

- java.lang.Object
- com.avoka.tm.svc.ErrorLogger

```
public class ErrorLogger
extends Object
```

Provides an Error Logger service which will log errors to the Transact error log database table.

Examples

Please find the email service examples about sending and queuing below.

Create Error Log Record For Transaction

The example below creates a Error Log record associated with the specified transaction record.

```
new ErrorLogger()
     .setTxnId(txnId)
     .setError(error)
     .log()
```

Create 'Form Delivery' Log Record For Transaction

The example below creates a 'Form Delivery' type error log record associated with the specified transaction record.

```
new ErrorLogger()
     .setError(error)
     .setType("Form Delivery")
     .setTxn(txn)
     .log()
```

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`ErrorLogger()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`Long`	`log()` Log the error creating an error log database record and return the error log record id.
`ErrorLogger`	`setError(Throwable error)` The error to log.
`ErrorLogger`	`setTxn(Txn txn)` Set the transaction to associate with the error log.
`ErrorLogger`	`setTxnId(Long txnId)`

	Set the transaction record id to associate with the error log.
`ErrorLogger`	`setType(String type)` Set the error log type property use to classify the error log record.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

ErrorLogger

```
public ErrorLogger()
```

Method Detail

setError

```
public ErrorLogger setError(Throwable error)
```

The error to log.

Parameters:

- `error` - the exception to log

Returns:

- the error logger;

---

setType

```
public ErrorLogger setType(String type)
```

Set the error log type property use to classify the error log record. Collaboration Job Email Form Delivery Form Dynamic Data Form Render Form Submission LiveCycle Management Payment Gateway Receipt Render REST SOAP API Scheduled Job Security Manager T.Field Sync Unclassified

Parameters:

- `type` - the error log record type (required)

Returns:

- the error logger

---

setTxn

```
public ErrorLogger setTxn(Txn txn)
```

Set the transaction to associate with the error log.

Parameters:

- `txn` - the transaction to associate with the error log

Returns:

- the error logger

---

setTxnId

```
public ErrorLogger setTxnId(Long txnId)
```

Set the transaction record id to associate with the error log.

Parameters:

- `txnId` - the transaction record id to associate with the error log

Returns:

- the error logger

---

log

```
public Long log()
```

Log the error creating an error log database record and return the error log record id.

Returns:

- the new error log record id

EventLogger

Package:

- com.avoka.tm.svc

Class EventLogger

- java.lang.Object
- com.avoka.tm.svc.EventLogger

```
public class EventLogger
extends Object
```

Provides an Event Logger service which will log [INFO, WARN | ERROR | SECURITY] events to the Transact event log database table.

Event Log records are not encrypted and should not contain sensitive PII data.

Examples

Please find the event logger examples below.

Create Info Event Log Record

The example below creates a 'Info' Event Log record associated with the specified transaction record.

```
import com.avoka.tm.svc.*

new EventLogger()
     .setMessage("IDV-Status - application completed")
     .setTxn(txn)
     .logInfo()
```

Create Security Event Log Record

The example below creates a 'Security' Event Log record associated with the specified transaction record. This example configures the EventLogger not to write the sensitive event log message to the server.log file.

```
import com.avoka.tm.svc.*

new EventLogger()
     .setMessage("Fraud Event with application : " + txn.formStatus + " : " + txn.formDataMap)
     .setRequest(request)
     .setTxn(txn)
     .setWriteToLogger(false)
     .logSecurity()
```

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`EventLogger()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`void`	`logError()` Log an Error event log record.
`void`	`logInfo()`

	Log an Information event log record.
void	logSecurity() Log an Security event log record.
void	logWarning() Log an Warning event log record.
EventLogger	setMessage(String message) Set the event log message property.
EventLogger	setRequest(HttpServletRequest request) Set the request property to log.
EventLogger	setTxn(Txn txn) Set the transaction to associate with the event log.
EventLogger	setTxnId(Long txnId) Set the transaction record id to associate with the event log.
EventLogger	setWriteToLogger(boolean writeToLogger) Specify whether to write a message to the Log4J server.log file.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

EventLogger

```
public EventLogger()
```

Method Detail

setMessage

```
public EventLogger setMessage(String message)
```

Set the event log message property.

Parameters:

- message - the message to log

Returns:

- the event logger

---

setRequest

```
public EventLogger setRequest(HttpServletRequest request)
```

Set the request property to log.

Parameters:

- request - the message to log

Returns:

- the event logger

---

setTxn

```
public EventLogger setTxn(Txn txn)
```

Set the transaction to associate with the event log.

Parameters:

- txn - the transaction to associate with the event log

Returns:

- the event logger

---

setTxnId

`public EventLogger setTxnId(Long txnId)`

Set the transaction record id to associate with the event log.

Parameters:

- `txnId` - the transaction record id to associate with the event log

Returns:

- the event logger

---

setWriteToLogger

`public EventLogger setWriteToLogger(boolean writeToLogger)`

Specify whether to write a message to the Log4J server.log file.

Parameters:

- `writeToLogger` - specify whether to write a message to the Log4J server.log file

Returns:

- the event logger

---

logInfo

`public void logInfo()`

Log an Information event log record.

---

logWarning

`public void logWarning()`

Log an Warning event log record.

---

logError

`public void logError()`

Log an Error event log record.

---

logSecurity

`public void logSecurity()`

Log an Security event log record.

GroovyServiceInvoker

Package:

- com.avoka.tm.svc

Class GroovyServiceInvoker

- java.lang.Object
- com.avoka.tm.svc.GroovyServiceInvoker

```
public class GroovyServiceInvoker
extends Object
```

Provides a 'Groovy Service' type service invoker class.

Please note Fluent Service can only invoke Dynamic Groovy Services if the install time Groovy Secure API restriction is not enforced.

Examples

Please find the groovy service invoker examples below.

Invoke "Reference Lookup" Service

The example below invokes a "Reference Lookup" Groovy Service with a map of parameters and returns a result object.

```
import com.avoka.tm.svc.*

def params = [:]
params.refNumber = ...

def result = new GroovyServiceInvoker()
     .setServiceName("Ref Lookup")
     .setClientCode("maguire")
     .setVersionNumber(2)
     .invoke(params)
```

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`GroovyServiceInvoker()` Create a groovy service invoker object.
`GroovyServiceInvoker(SvcDef svcDef)` Create a 'Groovy Service' invoker object.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`Object`	`invoke()` Invoke the service specified by the service definition name, version and client code and return the resulting object.
`Object`	`invoke(Map<String,Object> params)` Invoke the service specified by the service definition name, version and client code, with the given params and return the resulting object.
`GroovyServiceInvoker`	`setClientCode(String clientCode)` Set the service definition organization client code to invoke.

GroovyServiceInvoker	setServiceName(String serviceName) Set the service definition name to invoke.
GroovyServiceInvoker	setSvcDef(SvcDef svcDef) Set the service definition to invoke.
GroovyServiceInvoker	setVersionNumber(int versionNumber) Set the service definition version number to invoke.
GroovyServiceInvoker	withHighestVersion() Set flag to invoke the service with highest version number.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

GroovyServiceInvoker

```
public GroovyServiceInvoker()
```

Create a groovy service invoker object.

GroovyServiceInvoker

```
public GroovyServiceInvoker(SvcDef svcDef)
```

Create a 'Groovy Service' invoker object.

Parameters:

- svcDef - the 'Groovy Service' type service definition to invoke (required)

Method Detail

setSvcDef

```
public GroovyServiceInvoker setSvcDef(SvcDef svcDef)
```

Set the service definition to invoke.

Parameters:

- svcDef - the service definition

Returns:

- the groovy service invoker

setServiceName

```
public GroovyServiceInvoker setServiceName(String serviceName)
```

Set the service definition name to invoke.

Parameters:

- serviceName - the service definition name to invoke

Returns:

- the groovy service invoker

setVersionNumber

```
public GroovyServiceInvoker setVersionNumber(int versionNumber)
```

Set the service definition version number to invoke.

Parameters:

- versionNumber - service definition version number to invoke

Returns:

- the groovy service invoker

setClientCode

```
public GroovyServiceInvoker setClientCode(String clientCode)
```

Set the service definition organization client code to invoke.

Parameters:

- `clientCode` - service definition client code to invoke

Returns:

- the groovy service invoker

---

withHighestVersion

```
public GroovyServiceInvoker withHighestVersion()
```

Set flag to invoke the service with highest version number.

Returns:

- the service invoker

---

invoke

```
public Object invoke(Map<String,Object> params)
```

Invoke the service specified by the service definition name, version and client code, with the given params and return the resulting object.

Parameters:

- `params` - the map of parameter to invoke the groovy service with

Returns:

- invoke the service and return the resulting object

---

invoke

```
public Object invoke()
```

Invoke the service specified by the service definition name, version and client code and return the resulting object.

Returns:

- invoke the service and return the resulting object

PropertyBuilder

Package:

- com.avoka.tm.svc

Class PropertyBuilder

- java.lang.Object
- com.avoka.tm.svc.PropertyBuilder

```
public class PropertyBuilder
extends Object
```

Provides a property value builder service.

Examples

Example scripts are provided below.

Create/Update Form Property

The example below will create or update a "Product Codes" form version property against specified forms current form version. The property value may be use in Form Prefill or Dynamic Data services.

```
import com.avoka.tm.svc.*

String value = "product codes data..."

new PropertyBuilder()
    .setName("Product Codes")
    .setValue(value)
    .setFormCode("PDS-12")
    .build()
```

Create/Update Organization Property

The example below will create or update a "Loan Types" Organization property.

```
import com.avoka.tm.svc.*

String value = "loan types data..."

new PropertyBuilder()
    .setName("Loan Types")
    .setValue(value)
    .setClientCode("maguire")
    .build()
```

Create/Update Form Space Property

The example below will create or update a "Locale Messages" Form Space property.

```
import com.avoka.tm.svc.*

String value = "locale messages data..."

new PropertyBuilder()
    .setName("Locale Messages")
    .setValue(value)
    .setSpaceName("Work Space")
    .build()
```

Reference Data Loading Example

The example below is show a Fluent Groovy service provide a REST service endpoint for uploading an Orgs "Product Catalog" reference data set via a multi-part post request. This Fluent Groovy service is exposed via the REST Groovy Service Invoker API.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*
import javax.servlet.http.*
import org.apache.commons.fileupload.FileItem

class FluentGroovyService {

    Object invoke(SvcDef svcDef, HttpServletRequest request, User user, Map params) {

        // Get the products file upload file item
        FileItem fileItem = (FileItem) params.products

        logger.info "loading file: " + fileItem.name

        // Convert the fileItem binary data to CSV text
        String productsCSV = new String(fileItem.get(), "UTF-8")

        // Create or update the Organizations "Product Catalog" property
        new PropertyBuilder()
            .setName("Product Catalog")
            .setValue(productsCSV)
            .setClientCode(svcDef.clientCode)
            .build()

        String msg = "Imported '" + fileItem.name + "' file into 'Product Catalog' organization property"
        logger.info msg

        return msg
    }
}
```

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`PropertyBuilder()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`void`	`build()` Create or update the property based on the specified parameter.
`PropertyBuilder`	`setClientCode(String clientCode)` Set the property organization client code parameter.
`PropertyBuilder`	`setFormCode(String formCode)` Set the property form code parameter.
`PropertyBuilder`	`setName(String name)` Set the property name parameter.
`PropertyBuilder`	`setSpaceName(String spaceName)` Set the property space name parameter.
`PropertyBuilder`	`setValue(String value)` Set the property value parameter.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

PropertyBuilder

```
public PropertyBuilder()
```

Method Detail

setName

```
public PropertyBuilder setName(String name)
```

Set the property name parameter.

Parameters:

- `name` - the property name

Returns:

- the property builder

---

setValue

```
public PropertyBuilder setValue(String value)
```

Set the property value parameter.

Parameters:

- `value` - the property value

Returns:

- the property builder

---

setFormCode

```
public PropertyBuilder setFormCode(String formCode)
```

Set the property form code parameter.

Parameters:

- `formCode` - the property form code

Returns:

- the property builder

---

setClientCode

```
public PropertyBuilder setClientCode(String clientCode)
```

Set the property organization client code parameter.

Parameters:

- `clientCode` - the property organization client code

Returns:

- the property builder

---

setSpaceName

```
public PropertyBuilder setSpaceName(String spaceName)
```

Set the property space name parameter.

Parameters:

- `spaceName` - the property space name

Returns:

- the property builder

build

```
public void build()
```

Create or update the property based on the specified parameter.

ReceiptSvc

Package:

- com.avoka.tm.svc

Class ReceiptSvc

- java.lang.Object
- com.avoka.tm.svc.ReceiptSvc

```
public class ReceiptSvc
extends Object
```

Provides a PDF Receipt service which supports rendering PDF receipts on demand.

Examples

The example below ensures the form PDF receipt has been generated.

```
import com.avoka.tm.query.*
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

// Get the transaction object
Txn txn = ...

// Ensure PDF receipt has been rendered
new ReceiptSvc()
    .setTxn(txn)
    .renderReceipt()

// Re-query Txn and fetch PDF receipt data
txn = new TxnQuery()
    .setTxn(txn)
    .withReceiptPdf()
    .firstValue()

byte[] receiptPdf = txn.receiptPdf
```

Since:

- 5.0.1

Constructor Summary

Constructors

Constructor and Description
`ReceiptSvc()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`byte[]`	`renderReceipt()` Render the transaction PDF receipt and return the PDF bytes data.
`ReceiptSvc`	`setTxn(Txn txn)` Set the transaction parameter.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

ReceiptSvc

```
public ReceiptSvc()
```

Method Detail

setTxn

```
public ReceiptSvc setTxn(Txn txn)
```

Set the transaction parameter.

Parameters:

- `txn` - the transaction to set

Returns:

- the receipt service

---

renderReceipt

```
public byte[] renderReceipt()
```

Render the transaction PDF receipt and return the PDF bytes data.

Returns:

- the rendered PDF data

ServiceInvoker

Package:

- com.avoka.tm.svc

Class ServiceInvoker

- java.lang.Object
- com.avoka.tm.svc.ServiceInvoker

```
public class ServiceInvoker
extends Object
```

Provides a Service Definition service invoker class.

Examples

Please find the service invoker examples below.

Service Invoke Example

These Groovy examples show how to invoke service by name, client and version (current, highest, third).

```
import com.avoka.tm.svc.ServiceInvoker

Object currentVer = new ServiceInvoker()
            .setServiceName("myServiceName")
            .setClientCode("client")
            .invoke();

Object highestVer = new ServiceInvoker()
            .setServiceName("myServiceName")
            .setClientCode("client")
            .withHighestVersion()
            .invoke();

Object thirdVer = new ServiceInvoker()
            .setServiceName("myServiceName")
            .setClientCode("client")
            .setVersionNumber(3)
            .invoke()
```

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`ServiceInvoker()` Create a service invoker object.
`ServiceInvoker(SvcDef svcDef)` Create a service invoker object.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`Object`	`invoke()` Invoke the service specified by the service definition name, version and client code and return the resulting object.
`Object`	`invoke(Map<String,Object> params)`

	Invoke the service specified by the service definition name, version and client code, with the given params and return the resulting object.
ServiceInvoker	setClientCode(String clientCode) Set the service definition organization client code to invoke.
ServiceInvoker	setServiceName(String serviceName) Set the service definition name to invoke.
ServiceInvoker	setSvcDef(SvcDef svcDef) Set the service definition to invoke.
ServiceInvoker	setVersionNumber(int versionNumber) Set the service definition version number to invoke.
ServiceInvoker	withHighestVersion() Set flag to invoke the service with highest version number.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ServiceInvoker

```
public ServiceInvoker()
```

Create a service invoker object.

ServiceInvoker

```
public ServiceInvoker(SvcDef svcDef)
```

Create a service invoker object.

Parameters:

- svcDef - the service definition to invoke (required)

Method Detail

setSvcDef

```
public ServiceInvoker setSvcDef(SvcDef svcDef)
```

Set the service definition to invoke.

Parameters:

- svcDef - the service definition

Returns:

- the service invoker

setServiceName

```
public ServiceInvoker setServiceName(String serviceName)
```

Set the service definition name to invoke.

Parameters:

- serviceName - the service definition name to invoke

Returns:

- the service invoker

setVersionNumber

```
public ServiceInvoker setVersionNumber(int versionNumber)
```

Set the service definition version number to invoke. If not specified, then the current one will be used

Parameters:

- `versionNumber` - service definition version number to invoke

Returns:

- the service invoker

---

setClientCode

```
public ServiceInvoker setClientCode(String clientCode)
```

Set the service definition organization client code to invoke.

Parameters:

- `clientCode` - service definition client code to invoke

Returns:

- the service invoker

---

withHighestVersion

```
public ServiceInvoker withHighestVersion()
```

Set flag to invoke the service with highest version number.

Returns:

- the service invoker

---

invoke

```
public Object invoke(Map<String,Object> params)
```

Invoke the service specified by the service definition name, version and client code, with the given params and return the resulting object.

Parameters:

- `params` - the map of parameter to invoke the groovy service with

Returns:

- invoke the service and return the resulting object

---

invoke

```
public Object invoke()
```

Invoke the service specified by the service definition name, version and client code and return the resulting object.

Returns:

- invoke the service and return the resulting object

SvcConnUpdater

Package:

- com.avoka.tm.svc

Class SvcConnUpdater

- java.lang.Object
- com.avoka.tm.svc.SvcConnUpdater

```
public class SvcConnUpdater
extends Object
```

Provides a service connection value object updater class.

Examples

Please find the service connection updater examples below.

Update Service Connection Password

This Groovy example shows how to update a service connection password.

```
import com.avoka.tm.svc.*

SvcConn svcConn = ...

String password = "MyLittlePoney95"

new SvcConnUpdater(svcConn)
    .setPassword("param1")
    .update()
```

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`SvcConnUpdater()` Create a service connection updater.
`SvcConnUpdater(SvcConn svcConn)` Create a a service connection updater for the given svcConn value object.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`SvcConnUpdater`	`setEndpoint(String endpoint)` Set the service connection's endpoint to update.
`SvcConnUpdater`	`setFileData(byte[] filedata)` Set the service connection's file data to update.
`SvcConnUpdater`	`setFileName(String filename)` Set the service connection's file name to update.
`SvcConnUpdater`	`setId(Long id)` Set the service connection's id (PK) to identify the service connection to update.
`SvcConnUpdater`	`setParam1(String param1)`

	Set the service connection's param1 to update.
SvcConnUpdater	setParam2(String param2) Set the service connection's param2 to update.
SvcConnUpdater	setParam3(String param3) Set the service connection's param3 to update.
SvcConnUpdater	setParam4(String param4) Set the service connection's param4 to update.
SvcConnUpdater	setPassword(String password) Set the service connection's password to update.
SvcConnUpdater	setUsername(String username) Set the service connection's username to update.
void	update() Update the identified service connection with the specified parameters.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SvcConnUpdater

```
public SvcConnUpdater()
```

Create a service connection updater.

SvcConnUpdater

```
public SvcConnUpdater(SvcConn svcConn)
```

Create a a service connection updater for the given svcConn value object.

Parameters:

- svcConn - the service connection object to update (required)

Method Detail

setId

```
public SvcConnUpdater setId(Long id)
```

Set the service connection's id (PK) to identify the service connection to update.

Parameters:

- id - the service connection's id to identify the service connection to update (required)

Returns:

- the service connection updater

setUsername

```
public SvcConnUpdater setUsername(String username)
```

Set the service connection's username to update.

Parameters:

- username - the service connection's username to update

Returns:

- the service connection updater

setPassword

```
public SvcConnUpdater setPassword(String password)
```

Set the service connection's password to update.

Parameters:

- `password` - the service connection's password to update

Returns:

- the service connection updater

---

setEndpoint

```
public SvcConnUpdater setEndpoint(String endpoint)
```

Set the service connection's endpoint to update.

Parameters:

- `endpoint` - the service connection's endpoint to update

Returns:

- the service connection updater

---

setFileData

```
public SvcConnUpdater setFileData(byte[] filedata)
```

Set the service connection's file data to update.

Parameters:

- `filedata` - the service connection's file data to update

Returns:

- the service connection updater

---

setFileName

```
public SvcConnUpdater setFileName(String filename)
```

Set the service connection's file name to update.

Parameters:

- `filename` - the service connection's file name to update

Returns:

- the service connection updater

---

setParam1

```
public SvcConnUpdater setParam1(String param1)
```

Set the service connection's param1 to update.

Parameters:

- `param1` - the service connection's param1 to update

Returns:

- the service connection updater

---

setParam2

```
public SvcConnUpdater setParam2(String param2)
```

Set the service connection's param2 to update.

Parameters:

- `param2` - the service connection's param2 to update

Returns:

- the service connection updater

---

setParam3

`public SvcConnUpdater setParam3(String param3)`

Set the service connection's param3 to update.

Parameters:

- `param3` - the service connection's param3 to update

Returns:

- the service connection updater

---

setParam4

`public SvcConnUpdater setParam4(String param4)`

Set the service connection's param4 to update.

Parameters:

- `param4` - the service connection's param4 to update

Returns:

- the service connection updater

---

update

`public void update()`

Update the identified service connection with the specified parameters.

TrackingCodeBuilder

Package:

- com.avoka.tm.svc

Class TrackingCodeBuilder

- java.lang.Object
- com.avoka.tm.svc.TrackingCodeBuilder

```
public class TrackingCodeBuilder
extends Object
```

Provides a unique random tracking code builder.

The build method will ensure the tracking number is globally unique across both transaction and transaction history tables. If an existing tracking code is found, the build method will continue to generate a new tracking codes until it has created a globally unique value. If a unique value has not been generated after 100 attempts a `IllegalArgumentException` will be thrown.

Examples

Please find the tracking code builder examples below.

Create Transaction Tracking Code

The example below is creating an unique transaction tracking code and prefixing it with the form code and postfixing it with a test environment indicator.

```
class FluentTrackingNumberService {

    String invoke(SvcDef svcDef, Form form, HttpServletRequest request, User user) {

        return new TrackingCodeBuilder()
            .setPrefix(form.formCode + "-")
            .setPostfix("-TEST")
            .build()
    }

}
```

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`TrackingCodeBuilder()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`String`	`build()` Return a new random transaction tracking code (tracking number) for the specified properties.
`TrackingCodeBuilder`	`setPostfix(String postfix)` Set the tracking code value to postfix random component.
`TrackingCodeBuilder`	`setPrefix(String prefix)` Set the tracking code value to prefix random component.
`TrackingCodeBuilder`	`setRandomChars(String randomChars)` Set the character values to be used in the random component.
`TrackingCodeBuilder`	`setRandomLength(int randomLength)`

	Set the random component character length.
boolean	trackingCodeExists(String trackingCode) Return true if the specified tracking code is already used.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

TrackingCodeBuilder

```
public TrackingCodeBuilder()
```

Method Detail

setPrefix

```
public TrackingCodeBuilder setPrefix(String prefix)
```

Set the tracking code value to prefix random component.

Parameters:

- `prefix` - the tracking code prefix

Returns:

- the tracking code builder

---

setPostfix

```
public TrackingCodeBuilder setPostfix(String postfix)
```

Set the tracking code value to postfix random component.

Parameters:

- `postfix` - the tracking code post

Returns:

- the tracking code builder

---

setRandomLength

```
public TrackingCodeBuilder setRandomLength(int randomLength)
```

Set the random component character length. The default length is 7.

Parameters:

- `randomLength` - the random component character length

Returns:

- the tracking code builder

---

setRandomChars

```
public TrackingCodeBuilder setRandomChars(String randomChars)
```

Set the character values to be used in the random component. The default values include: BCDFGHJKLMNPQRSTVWXYZ23456789

Parameters:

- `randomChars` - the character values to be used in the random component.

Returns:

- the tracking code builder

---

trackingCodeExists

```
public boolean trackingCodeExists(String trackingCode)
```

Return true if the specified tracking code is already used.

Parameters:

- `trackingCode` - the tracking code value to check (must not be null or an empty string)

Returns:

- true if the tracking code value is already in use or there was an error during lookup

---

build

```
public String build()
```

Return a new random transaction tracking code (tracking number) for the specified properties.

This method will ensure the tracking number is globally unique across both transaction and transaction history tables. If an existing tracking code is found, this method will continue to generate a new tracking codes until a globally unique value is found. If this method cannot generate a unique value after 10 attempts it will throw an `IllegalArgumentException`

If a unique tracking code is found, the code is also stored in the tracking_number table that records all unique codes.

Returns:

- new random transaction tracking code (tracking number) for the specified properties.

TxnBuilder

Package:

- com.avoka.tm.svc

Class TxnBuilder

- java.lang.Object
- com.avoka.tm.svc.TxnBuilder

```
public class TxnBuilder
extends Object
```

Provides a Txn value object builder class.

This class builds anonymous saved form, task form or review form transactions. If `contactEmailAddress` is set, then it will be built anonymous saved form, otherwise specifying `loginName` and `taskType` with [Form|Review] value will determine what form type will be created.

Building form task transaction basically creates a form task transaction for a user or a user group to do. The task form can be merged and rendered either by the form schema seed or using the prefill data services. Note that form data XML and prefill data XML are mutually exclusive. Note that this method contains database transactions.

Building review task transaction basically creates a review task transaction for a user or a user group to do. A review task is based on an existing transaction and can be pre-filled with schema seed data.

Examples

Please find transaction builder examples below.

Create an Anonymous Saved Form

Build an anonymous saved form transaction using the provided parameters data.

```
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

Txn txn = new TxnBuilder()
             .setAddress("1 Street")
             .setContactEmailAddress("contact@email.com")
             .setDatetimeScheduled(new Date())
             .setDatetimeExpiry(new Date())
             .setEmailSubject("Email subject")
             .setEmailMessage("Email message")
             .setFormCode("FRM-1234")
             .setGroupName("Administrator")
             .setLatitude(new Double(40))
             .setLongitude(new Double(50))
             .setProperty("propKey", "propValue")
             .setReceiptNumber("RCPT-1234")
             .setSaveChallengeAnswer("saveChallengeAnswer")
             .setSequence(new Integer(2))
             .setSpaceName("spaceName")
             .setTxnXml("<Txn></Txn>")
             .setTaskMessage("taskMessage")
             .setTaskSubject("taskSubject")
             .setTransRefNumber("TRANS-1234")
             .withFormDataMap()
             .withGroupNames()
             .withPropertyMap()
             .build()
```

Create a Form Task Form

Build a form task transaction using the provided parameter data.

```
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

Txn txn = new TxnBuilder()
```

```
            .setAddress("1 Street")
            .setDatetimeExpiry(new Date())
            .setDatetimeScheduled(new Date())
            .setEmailSubject("Email subject")
            .setEmailMessage("Email message")
            .setFormDataXml("<FormData></FormData>")
            .setFormCode("FRM-1234")
            .setLatitude(new Double(40))
            .setLoginName("login")
            .setLongitude(new Double(50))
            .setSequence(new Integer(2))
            .setSpaceName("space")
            .setTaskMessage("taskMessage")
            .setTaskSubject("taskSubject")
            .setTaskType("Form")
            .withUserDeletableFlag()
            .build()
```

Create a Review Task

Build a review task transaction using the provided parameter data.

```
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

Txn txn = new TxnBuilder()
            .setLoginName("login")
            .setTaskType("Review")
            .setTaskSubject("Task subject")
            .setReviewTxn(new Long(12345678))
            .build()
```

Since:

- 5.0.0

See Also:

- Txn

Constructor Summary

Constructors

Constructor and Description
TxnBuilder()

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
TxnBuilder	addFileAttach(FileAttach fileAttach) Add the given file attachment to the transaction.
TxnBuilder	addFormDataExtract(String name, String value) Add form data (data extract) to the transaction.
Txn	build() Create a transaction based on the specified parameter and return the new Txn value object.
TxnBuilder	setAddress(String address) Set address parameter.
TxnBuilder	setContactEmailAddress(String contactEmailAddress) Set the contact email address parameter.
TxnBuilder	setCopyAttachmentsNames(List<String> copyAttachmentsNames) Set copy attachment names parameter.
TxnBuilder	setCopyAttachmentsTxn(Long copyAttachmentsTxn) Set copy attachments txn id parameter.

TxnBuilder	`setDatetimeExpiry(Date datetimeExpiry)` Set date time expiry parameter.
TxnBuilder	`setDatetimeScheduled(Date datetimeScheduled)` Set date time scheduled parameter.
TxnBuilder	`setEmailMessage(String emailMessage)` Set email message parameter.
TxnBuilder	`setEmailSubject(String emailSubject)` Set email subject parameter.
TxnBuilder	`setFormCode(String formCode)` Set the form code parameter.
TxnBuilder	`setFormDataXml(String formDataXml)` Set form data xml parameter.
TxnBuilder	`setFormPrefillServiceName(String formPrefillServiceName)` Set form prefill service name parameter.
TxnBuilder	`setGroupName(String groupName)` Set group name parameter.
TxnBuilder	`setGroupNames(Collection<String> groupNames)` Set group names parameter.
TxnBuilder	`setInputXmlData(String inputXmlData)` Set input xml data parameter.
TxnBuilder	`setJobAction(Long jobAction)` Set job action parameter.
TxnBuilder	`setLatitude(Double latitude)` Set latitude parameter.
TxnBuilder	`setLoginName(String loginName)` Set the login name parameter.
TxnBuilder	`setLongitude(Double longitude)` Set longiture parameter.
TxnBuilder	`setProperty(String name, String value)` Specify the property name and value on the transaction.
TxnBuilder	`setPropertyMap(Map<String,String> propertyMap)` Specify the property map on the transaction.
TxnBuilder	`setReceiptNumber(String receiptNumber)` Set receipt number parameter.
TxnBuilder	`setReviewTxn(Long reviewTxnId)` Set the review txn id parameter.
TxnBuilder	`setSaveChallengeAnswer(String saveChallengeAnswer)` Set save challenge answer parameter.
TxnBuilder	`setSequence(Integer sequence)` Set sequence number parameter.
TxnBuilder	`setSpaceName(String spaceName)` Set the space name parameter.
TxnBuilder	`setTaskMessage(String taskMessage)` Set task message parameter.
TxnBuilder	`setTaskSubject(String taskSubject)` Set the task subject parameter.
TxnBuilder	`setTaskType(String taskType)` Set the task type parameter [Form | Review].
TxnBuilder	`setTransRefNumber(String transRefNumber)` Set transaction reference number parameter.

TxnBuilder	setTxnXml(String txnXml) Set txn xml parameter.
TxnBuilder	withAllowClaimFlag() Set allow claim flag parameter to true.
TxnBuilder	withFormDataMap() Set the query to return the transaction with the associated form data map information.
TxnBuilder	withGroupNames() Set the query to return the transaction with the associated group names information.
TxnBuilder	withPropertyMap() Set the query to return the transaction with the associated property map information.
TxnBuilder	withSendEmailFlag() Set send email flag parameter to true.
TxnBuilder	withUserDeletableFlag() Set user deletable flag parameter to true.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

TxnBuilder

```
public TxnBuilder()
```

Method Detail

setSpaceName

```
public TxnBuilder setSpaceName(String spaceName)
```

Set the space name parameter.

Parameters:

- spaceName - the space name parameter

Returns:

- the transaction builder

---

setContactEmailAddress

```
public TxnBuilder setContactEmailAddress(String contactEmailAddress)
```

Set the contact email address parameter.

Parameters:

- contactEmailAddress - the email address parameter

Returns:

- the transaction builder

---

setFormCode

```
public TxnBuilder setFormCode(String formCode)
```

Set the form code parameter.

Parameters:

- formCode - the form code parameter

Returns:

- the transaction builder

---

setLoginName

`public TxnBuilder setLoginName(String loginName)`

Set the login name parameter.

Parameters:

- `loginName` - the login name parameter

Returns:

- the transaction builder

---

setTaskType

`public TxnBuilder setTaskType(String taskType)`

Set the task type parameter [Form | Review].

Parameters:

- `taskType` - the task type parameter [Form | Review]

Returns:

- the transaction builder

---

setReviewTxn

`public TxnBuilder setReviewTxn(Long reviewTxnId)`

Set the review txn id parameter.

Parameters:

- `reviewTxnId` - the review txn id

Returns:

- the transaction builder

---

setTaskSubject

`public TxnBuilder setTaskSubject(String taskSubject)`

Set the task subject parameter.

Parameters:

- `taskSubject` - task subject

Returns:

- the transaction builder

---

setGroupName

`public TxnBuilder setGroupName(String groupName)`

Set group name parameter.

Parameters:

- `groupName` - group name

Returns:

- the transaction builder

---

setAddress

`public TxnBuilder setAddress(String address)`

Set address parameter.

Parameters:

- `address` - address

Returns:

- the transaction builder

---

setCopyAttachmentsNames

```
public TxnBuilder setCopyAttachmentsNames(List<String> copyAttachmentsNames)
```

Set copy attachment names parameter.

Parameters:

- `copyAttachmentsNames` - copy attachment names

Returns:

- the transaction builder

---

setCopyAttachmentsTxn

```
public TxnBuilder setCopyAttachmentsTxn(Long copyAttachmentsTxn)
```

Set copy attachments txn id parameter.

Parameters:

- `copyAttachmentsTxn` - copy attachments txn id

Returns:

- the transaction builder

---

setEmailSubject

```
public TxnBuilder setEmailSubject(String emailSubject)
```

Set email subject parameter.

Parameters:

- `emailSubject` - email subject

Returns:

- the transaction builder

---

setEmailMessage

```
public TxnBuilder setEmailMessage(String emailMessage)
```

Set email message parameter.

Parameters:

- `emailMessage` - email message

Returns:

- the transaction builder

---

setDatetimeScheduled

```
public TxnBuilder setDatetimeScheduled(Date datetimeScheduled)
```

Set date time scheduled parameter.

Parameters:

- `datetimeScheduled` - date time scheduled

Returns:

- the transaction builder

---

setDatetimeExpiry

```
public TxnBuilder setDatetimeExpiry(Date datetimeExpiry)
```

Set date time expiry parameter.

Parameters:

- `datetimeExpiry` - date time expiry

Returns:

- the transaction builder

---

setFormPrefillServiceName

```
public TxnBuilder setFormPrefillServiceName(String formPrefillServiceName)
```

Set form prefill service name parameter.

Parameters:

- `formPrefillServiceName` - form prefill service name

Returns:

- the transaction builder

---

setInputXmlData

```
public TxnBuilder setInputXmlData(String inputXmlData)
```

Set input xml data parameter.

Parameters:

- `inputXmlData` - input xml data

Returns:

- the transaction builder

---

setLatitude

```
public TxnBuilder setLatitude(Double latitude)
```

Set latitude parameter.

Parameters:

- `latitude` - latitude parameter

Returns:

- the transaction builder

---

setLongitude

```
public TxnBuilder setLongitude(Double longitude)
```

Set longiture parameter.

Parameters:

- `longitude` - longitude parameter

Returns:

- the transaction builder

---

setReceiptNumber

```
public TxnBuilder setReceiptNumber(String receiptNumber)
```

Set receipt number parameter.

Parameters:

- `receiptNumber` - receipt number

Returns:

- the transaction builder

setSaveChallengeAnswer

`public TxnBuilder setSaveChallengeAnswer(String saveChallengeAnswer)`

Set save challenge answer parameter.

Parameters:

- `saveChallengeAnswer` - save challenge answer

Returns:

- the transaction builder

---

setSequence

`public TxnBuilder setSequence(Integer sequence)`

Set sequence number parameter.

Parameters:

- `sequence` - sequnce number

Returns:

- the transaction builder

---

withSendEmailFlag

`public TxnBuilder withSendEmailFlag()`

Set send email flag parameter to `true`.

Returns:

- the transaction builder

---

setTxnXml

`public TxnBuilder setTxnXml(String txnXml)`

Set txn xml parameter.

Parameters:

- `txnXml` - txn xml

Returns:

- the transaction builder

---

setTaskMessage

`public TxnBuilder setTaskMessage(String taskMessage)`

Set task message parameter.

Parameters:

- `taskMessage` - task message

Returns:

- the transaction builder

---

setTransRefNumber

`public TxnBuilder setTransRefNumber(String transRefNumber)`

Set transaction reference number parameter.

Parameters:

- `transRefNumber` - transaction reference number

Returns:

- the transaction builder

---

setGroupNames

```
public TxnBuilder setGroupNames(Collection<String> groupNames)
```

Set group names parameter.

Parameters:

- `groupNames` - group names

Returns:

- the transaction builder

---

setJobAction

```
public TxnBuilder setJobAction(Long jobAction)
```

Set job action parameter.

Parameters:

- `jobAction` - job action

Returns:

- the transaction builder

---

setProperty

```
public TxnBuilder setProperty(String name,
                              String value)
```

Specify the property name and value on the transaction.

Parameters:

- `name` - the property name (required)
- `value` - the property value (required)

Returns:

- the transaction builder

Since:

- 5.1.0

---

setPropertyMap

```
public TxnBuilder setPropertyMap(Map<String,String> propertyMap)
```

Specify the property map on the transaction.

Parameters:

- `propertyMap` - the property map

Returns:

- the transaction builder

Since:

- 5.1.0

---

addFileAttach

```
public TxnBuilder addFileAttach(FileAttach fileAttach)
```

Add the given file attachment to the transaction.

Parameters:

- `fileAttach` - the file attachment to add to the transaction

Returns:

- the transaction builder

---

withAllowClaimFlag

```
public TxnBuilder withAllowClaimFlag()
```

Set allow claim flag parameter to `true`.

Returns:

- the transaction builder

---

setFormDataXml

```
public TxnBuilder setFormDataXml(String formDataXml)
```

Set form data xml parameter.

Parameters:

- `formDataXml` - the form data XML

Returns:

- the transaction builder

---

addFormDataExtract

```
public TxnBuilder addFormDataExtract(String name,
                                     String value)
```

Add form data (data extract) to the transaction.

Parameters:

- `name` - form data (extract) name
- `value` - form data (extract) value

Returns:

- the transaction builder

---

withUserDeletableFlag

```
public TxnBuilder withUserDeletableFlag()
```

Set user deletable flag parameter to `true`.

Returns:

- the transaction builder

---

withFormDataMap

```
public TxnBuilder withFormDataMap()
```

Set the query to return the transaction with the associated form data map information.

Returns:

- the transaction builder

---

withGroupNames

```
public TxnBuilder withGroupNames()
```

Set the query to return the transaction with the associated group names information.

Returns:

- the transaction builder

---

withPropertyMap

```
public TxnBuilder withPropertyMap()
```

Set the query to return the transaction with the associated property map information.

Returns:

- the transaction builder

Since:

- 5.1.0

---

build

```
public Txn build()
```

Create a transaction based on the specified parameter and return the new Txn value object.

Returns:

- the newly created Txn value object

TxnCheckpointSvc

Package:

- com.avoka.tm.svc

Class TxnCheckpointSvc

- java.lang.Object
- com.avoka.tm.svc.TxnCheckpointSvc

```
public class TxnCheckpointSvc
extends Object
```

Provides a transaction delivery checkpoint service.

Examples

Please find the transaction checkpoint examples below.

Transaction Checkpoint Example

This Groovy example shows if transaction check point should be performed.

```
import com.avoka.tm.svc.TxnCheckpointSvc

boolean neededCheckpoint = new TxnCheckpointSvc()
                           .doCheckpoint("checkpoint")
```

Transaction Complete Checkpoint Example

This Groovy example shows how to add Completed the delivery checkpoint with the given name, if it does not already exists.

```
import com.avoka.tm.svc.TxnCheckpointSvc;

boolean addedCheckpoint = new TxnCheckpointSvc()
                          .complete("checkpoint")
```

Check for existence of 'Completed' checkpoint Example

This Groovy example shows how to see whether a 'Completed' checkpoint with the specified name already exists for the submission.

```
import com.avoka.tm.svc.TxnCheckpointSvc;

boolean addedCheckpoint = new TxnCheckpointSvc()
                          .complete("checkpoint")
```

Since:

- 5.0

Constructor Summary

Constructors

Constructor and Description
`TxnCheckpointSvc(Txn txn)` Create a transaction delivery checkpoint service for the given transaction.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
void	clearAll() Deletes all existing delivery checkpoints for the configured submission.
boolean	complete(String name) Add a Completed the delivery checkpoint with the given name, if it does not already exists.
boolean	complete(String name, String description) Add a Completed the delivery checkpoint with the given name, if it does not already exists.
boolean	doCheckpoint(String name) Return true if the checkpoint should be performed, either hasn't been registered or an error has occurred previously.
boolean	doCheckpoint(String name, String description) Return true if the checkpoint should be performed, either hasn't been registered or an error has occurred previously.
void	error(String name, Object error) Add a Error the delivery checkpoint with the given name, if it does not already exists.
boolean	isCompleted(String name) Return whether a 'Completed' checkpoint with the specified name already exists for the submission.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

TxnCheckpointSvc

```
public TxnCheckpointSvc(Txn txn)
```

Create a transaction delivery checkpoint service for the given transaction.

Parameters:

- txn - the transaction value object (required)

Method Detail

doCheckpoint

```
public boolean doCheckpoint(String name)
```

Return true if the checkpoint should be performed, either hasn't been registered or an error has occurred previously.

If the checkpoint has not already been registered this method will create a new checkpoint.

Parameters:

- name - the name of the checkpoint to check, and register if it doesn't exist (required)

Returns:

- if the checkpoint needs to be performed or false if already completed

---

doCheckpoint

```
public boolean doCheckpoint(String name,
                            String description)
```

Return true if the checkpoint should be performed, either hasn't been registered or an error has occurred previously.

If the checkpoint has not already been registered this method will create a new checkpoint.

Parameters:

- name - the name of the checkpoint to check, and register if it doesn't exist (required)
- description - the checkpoint description (optional)

Returns:

- if the checkpoint needs to be performed or false if already completed

Since:

- 4.2.0

isCompleted

```
public boolean isCompleted(String name)
```

Return whether a 'Completed' checkpoint with the specified name already exists for the submission.

Parameters:

- `name` - the checkpoint name (required)

Returns:

- true if the 'Completed' checkpoint already exists

---

complete

```
public boolean complete(String name)
```

Add a Completed the delivery checkpoint with the given name, if it does not already exists.

Parameters:

- `name` - the checkpoint name (required)

Returns:

- true if the new completed delivery checkpoint was added or false if the checkpoint was already present

---

complete

```
public boolean complete(String name,
                        String description)
```

Add a Completed the delivery checkpoint with the given name, if it does not already exists.

Parameters:

- `name` - the checkpoint name (required)
- `description` - the delivery checkpoint description (optional)

Returns:

- true if the new completed delivery checkpoint was added or false if the checkpoint was already present

---

error

```
public void error(String name,
                  Object error)
```

Add a Error the delivery checkpoint with the given name, if it does not already exists. If the passed in error object is an `Throwable` error it will be logged to the Error Log and associated with the Submission.

Parameters:

- `name` - the checkpoint name (required)
- `error` - message the delivery checkpoint error message (optional)

---

clearAll

```
public void clearAll()
```

Deletes all existing delivery checkpoints for the configured submission. This method performs a database commit.

TxnUpdater

Package:

- com.avoka.tm.svc

Class TxnUpdater

- java.lang.Object
- com.avoka.tm.svc.TxnUpdater

```
public class TxnUpdater
extends Object
```

Provides a transaction value object updater class.

Examples

Please find the transaction updater examples below.

Set Transaction Delivery Ready

This Groovy example shows how to update a transaction setting its delivery status to 'Ready'.

```
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

new TxnUpdater(txn)
    .setDeliveryStatus(Txn.DELIVERY_READY)
    .update()
```

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`TxnUpdater()` Create a transaction updater.
`TxnUpdater(Txn txn)` Create a transaction updater for the given txn value object.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`TxnUpdater`	`addFileAttach(FileAttach fileAttach)` Add the given file attachment to the transaction.
`TxnUpdater`	`addFormDataExtract(String name, String value)` Add form data (data extract) to the transaction.
`TxnUpdater`	`addServiceCallLog(String svcName, String info, String url)` Add a service call log to the transaction.
`TxnUpdater`	`removeFileAttach(FileAttach fileAttach)` Remove the given file attachment from the transaction.
`TxnUpdater`	`removeFormDataExtract(String name)` Remove form data (data extract) from the transaction.
`TxnUpdater`	`removeProperty(String name)`

	Specify the property to remove from the transaction.
TxnUpdater	setAttachmentsStatus(String attachmentsStatus) The attachments status value to set on the transaction.
TxnUpdater	setDeliveryChannel(String deliveryChannel) The organization delivery channel to associate with the transaction.
TxnUpdater	setDeliveryStatus(String deliveryStatus) The delivery status value to set on the transaction.
TxnUpdater	setExternalUserId(String externalUserId) Set external user id to the transaction.
TxnUpdater	setFormStatus(String formStatus) The form status value to set on the transaction.
TxnUpdater	setFormXml(String formXml) The form XML data value to set on the transaction.
TxnUpdater	setId(Number id) Set the transaction id (PK) to identify the transaction to update.
TxnUpdater	setPaymentStatus(String paymentStatus) The payment status value to set on the transaction.
TxnUpdater	setProperty(String name, String value) Specify the property name and value to add or update on the transaction.
TxnUpdater	setReceiptStatusReady() Set the transaction PDF receipt generation status to be ready.
TxnUpdater	setSaveChallengeAnswer(String saveChallengeAnswer) Set save challenge answer to the transaction.
TxnUpdater	setSaveChallengeTimeout(Date saveChallengeTimeout) Set save challenge timeout to the transaction.
TxnUpdater	setSpace(Space space) The space value to set on the transaction.
TxnUpdater	setUser(User user) The user value to set on the transaction.
TxnUpdater	setUserSaved(boolean userSaved) Set the user saved to the transaction.
void	update() Update the identified transaction with the specified parameters.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

TxnUpdater

```
public TxnUpdater()
```

Create a transaction updater.

TxnUpdater

```
public TxnUpdater(Txn txn)
```

Create a transaction updater for the given txn value object.

Parameters:

- txn - the transaction value object to update (required)

Method Detail

setId

`public TxnUpdater setId(Number id)`

Set the transaction id (PK) to identify the transaction to update.

Parameters:

- `id` - the transaction id of the transaction to update (required)

Returns:

- the transaction updater

---

setFormStatus

`public TxnUpdater setFormStatus(String formStatus)`

The form status value to set on the transaction.

Parameters:

- `formStatus` - the form status value to set on the transaction

Returns:

- the transaction updater

---

setAttachmentsStatus

`public TxnUpdater setAttachmentsStatus(String attachmentsStatus)`

The attachments status value to set on the transaction.

Parameters:

- `attachmentsStatus` - attachments status value on the transaction

Returns:

- the transaction updater

---

setPaymentStatus

`public TxnUpdater setPaymentStatus(String paymentStatus)`

The payment status value to set on the transaction.

Parameters:

- `paymentStatus` - the payment status value to set on the transaction

Returns:

- the transaction updater

---

setSpace

`public TxnUpdater setSpace(Space space)`

The space value to set on the transaction.

Parameters:

- `space` - the space value to set on the transaction

Returns:

- the transaction updater

Since:

- 5.0.3

---

setUser

`public TxnUpdater setUser(User user)`

The user value to set on the transaction.

Parameters:

- `user` - the user value to set on the transaction

Returns:

- the transaction updater

Since:

- 5.0.3

---

setReceiptStatusReady

```
public TxnUpdater setReceiptStatusReady()
```

Set the transaction PDF receipt generation status to be ready.

Returns:

- the transaction updater

---

setDeliveryStatus

```
public TxnUpdater setDeliveryStatus(String deliveryStatus)
```

The delivery status value to set on the transaction.

Parameters:

- `deliveryStatus` - the delivery status value to set on the transaction

Returns:

- the transaction updater

---

setDeliveryChannel

```
public TxnUpdater setDeliveryChannel(String deliveryChannel)
```

The organization delivery channel to associate with the transaction.

Parameters:

- `deliveryChannel` - the name of the organization delivery channel (required)

Returns:

- the transaction updater

---

setProperty

```
public TxnUpdater setProperty(String name,
                              String value)
```

Specify the property name and value to add or update on the transaction.

Parameters:

- `name` - the property name (required)
- `value` - the property value (required)

Returns:

- the transaction updater

---

removeProperty

```
public TxnUpdater removeProperty(String name)
```

Specify the property to remove from the transaction.

Parameters:

- `name` - the property name to remove (required)

Returns:

- the transaction updater

---

setFormXml

```
public TxnUpdater setFormXml(String formXml)
```

The form XML data value to set on the transaction.

Parameters:

- `formXml` - the form XML data value to set on the transaction (required)

Returns:

- the transaction updater

---

addFileAttach

```
public TxnUpdater addFileAttach(FileAttach fileAttach)
```

Add the given file attachment to the transaction.

Parameters:

- `fileAttach` - the file attachment to add to the transaction (required)

Returns:

- the transaction updater

---

removeFileAttach

```
public TxnUpdater removeFileAttach(FileAttach fileAttach)
```

Remove the given file attachment from the transaction.

Parameters:

- `fileAttach` - the file attachment to remove from the transaction (required)

Returns:

- the transaction updater

---

addServiceCallLog

```
public TxnUpdater addServiceCallLog(String svcName,
                                    String info,
                                    String url)
```

Add a service call log to the transaction.

Parameters:

- `svcName` - the service definition called (required)
- `info` - the service call information (optional)
- `url` - the service URL called (optional)

Returns:

- the transaction updater

---

setExternalUserId

```
public TxnUpdater setExternalUserId(String externalUserId)
```

Set external user id to the transaction.

Parameters:

- `externalUserId` - external user id

Returns:

- the transaction updater

---

addFormDataExtract

```
public TxnUpdater addFormDataExtract(String name,
                                     String value)
```

Add form data (data extract) to the transaction.

Parameters:

- `name` - form data (extract) name
- `value` - form data (extract) value

Returns:

- the transaction updater

---

removeFormDataExtract

```
public TxnUpdater removeFormDataExtract(String name)
```

Remove form data (data extract) from the transaction.

Parameters:

- `name` - form data (extract) name

Returns:

- the transaction updater

---

setSaveChallengeAnswer

```
public TxnUpdater setSaveChallengeAnswer(String saveChallengeAnswer)
```

Set save challenge answer to the transaction.

Parameters:

- `saveChallengeAnswer` - save challenge answer

Returns:

- the transaction updater

Since:

- 5.1.0

---

setSaveChallengeTimeout

```
public TxnUpdater setSaveChallengeTimeout(Date saveChallengeTimeout)
```

Set save challenge timeout to the transaction.

Parameters:

- `saveChallengeTimeout` - save challenge timeout

Returns:

- the transaction updater

Since:

- 5.1.0

---

setUserSaved

```
public TxnUpdater setUserSaved(boolean userSaved)
```

Set the user saved to the transaction.

Parameters:

- `userSaved` - the the user saved

Returns:

- the transaction updater

Since:

- 5.1.3

---

update

```
public void update()
```

Update the identified transaction with the specified parameters.

UserBuilder

Package:

- com.avoka.tm.svc

Class UserBuilder

- java.lang.Object
- com.avoka.tm.svc.UserBuilder

```
public class UserBuilder
extends Object
```

Provides a User value object builder class.

Examples

Please find the user builder examples below.

Create a User

This example shows how to create a User object.

```
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

User user = new UserBuilder("Work Space")
               .setEmail("my@email.com")
               .setUserType(User.TYPE_LOCAL)
               .setLoginName("login")
               .setPassword("secret!23")
               .create()
```

Update a User's Email Address

This example shows how to update an user's email address.

```
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

User user = new UserBuilder(request)
               .setLoginName("dsmith")
               .setEmail("dsmith@email.com")
               .update()
```

Create/Update a User

The example below shows how to create or update an 'SSO' type user.

```
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

User user = new UserBuilder(request)
               .setLoginName("dsmith")
               .setEmail("dsmith@email.com")
               .setUserType(User.TYPE_SSO)
               .createOrUpdate()
```

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`UserBuilder(HttpServletRequest request)` Create a User service object with the given request.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`UserBuilder`	`addGroupName(String groupName)` Add the group names the user should be assigned to.
`UserBuilder`	`addOrgName(String orgName)` Add the organization name the user should be associated with.
`UserBuilder`	`addRoleName(String roleName)` Add the role name the user should be assigned to.
`UserBuilder`	`addSpaceName(String spaceName)` Add the form space names the user should be assigned to.
`User`	`create()` Create a new User account based on the specified parameter and return the new User value object.
`User`	`createOrUpdate()` Create or update the user account specified by the given parameters, and return current User value object.
`UserBuilder`	`removeGroupName(String groupName)` Remove the group names the user should be assigned to.
`UserBuilder`	`removeOrgName(String orgName)` Remove the organization name the user should be associated with.
`UserBuilder`	`removeRoleName(String roleName)` Remove the role name the user should be assigned to.
`UserBuilder`	`removeSpaceName(String spaceName)` Remove the form space names the user should be assigned to.
`UserBuilder`	`setAccountStatus(String accountStatus)` Set the user account status parameter [Active | Inactive | Locked | Locked Temporarily | Pending | Rejected].
`UserBuilder`	`setEmail(String email)` Set the user email parameter.
`UserBuilder`	`setFirstName(String firstName)` Set the user first name parameter.
`UserBuilder`	`setGroupNames(Set<String> groupNames)` Set the group names the user should be assigned to.
`UserBuilder`	`setLastName(String lastName)` Set the user last name parameter.
`UserBuilder`	`setLoginName(String loginName)` Set the user login name parameter.
`UserBuilder`	`setMobile(String mobile)` Set the user mobile number parameter.
`UserBuilder`	`setOrgNames(Set<String> orgNames)` Set the organization names the user should be associated with.
`UserBuilder`	`setPassword(String password)` Set the user password, only applicable for "Local" type users and not for LDAP or SSO user types.
`UserBuilder`	`setProfileMap(Map<String,String> profileMap)` Set the user profile values map parameter.
`UserBuilder`	`setRoleNames(Set<String> roleNames)` Set the role names the user should be assigned to.

UserBuilder	setSpaceNames(Set<String> spaceNames) Set the form space names the user should be assigned to.
UserBuilder	setUserType(String userType) Set the user type parameter [Local | LDAP | SSO].
User	update() Update the user account specified by the loginName parameter and return the update User value object.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

UserBuilder

```
public UserBuilder(HttpServletRequest request)
```

Create a User service object with the given request.

Parameters:

- request - the servlet request

Method Detail

setLoginName

```
public UserBuilder setLoginName(String loginName)
```

Set the user login name parameter.

Parameters:

- loginName - the user login name parameter

Returns:

- the user service

---

setAccountStatus

```
public UserBuilder setAccountStatus(String accountStatus)
```

Set the user account status parameter [Active | Inactive | Locked | Locked Temporarily | Pending | Rejected].

Parameters:

- accountStatus - the user account status parameter [Active | Inactive | Locked | Locked Temporarily | Pending | Rejected]

Returns:

- the user service

---

setEmail

```
public UserBuilder setEmail(String email)
```

Set the user email parameter.

Parameters:

- email - the user email parameter

Returns:

- the user service

---

setFirstName

```
public UserBuilder setFirstName(String firstName)
```

Set the user first name parameter.

Parameters:

- `firstName` - the user first name parameter

Returns:

- the user service

---

setLastName

```
public UserBuilder setLastName(String lastName)
```

Set the user last name parameter.

Parameters:

- `lastName` - the user last name parameter

Returns:

- the user service

---

setMobile

```
public UserBuilder setMobile(String mobile)
```

Set the user mobile number parameter.

Parameters:

- `mobile` - the user mobile number parameter

Returns:

- the user service

---

setPassword

```
public UserBuilder setPassword(String password)
```

Set the user password, only applicable for "Local" type users and not for LDAP or SSO user types.

Parameters:

- `password` - the user password to set

Returns:

- the user service

---

setUserType

```
public UserBuilder setUserType(String userType)
```

Set the user type parameter [Local | LDAP | SSO].

Parameters:

- `userType` - the user type parameter [Local | LDAP | SSO]

Returns:

- the user service

---

setProfileMap

```
public UserBuilder setProfileMap(Map<String,String> profileMap)
```

Set the user profile values map parameter.

Parameters:

- `profileMap` - the user profile values map parameter

Returns:

- the user service

---

setGroupNames

`public UserBuilder setGroupNames(Set<String> groupNames)`

Set the group names the user should be assigned to.

Parameters:

- `groupNames` - the group names the user should be assigned to

Returns:

- the user service

---

addGroupName

`public UserBuilder addGroupName(String groupName)`

Add the group names the user should be assigned to.

Parameters:

- `groupName` - the group name the user should be assigned to

Returns:

- the user service

---

removeGroupName

`public UserBuilder removeGroupName(String groupName)`

Remove the group names the user should be assigned to.

Parameters:

- `groupName` - the group name the user should not be assigned to

Returns:

- the user service

---

setOrgNames

`public UserBuilder setOrgNames(Set<String> orgNames)`

Set the organization names the user should be associated with.

Parameters:

- `orgNames` - the organization names the user should be associated with

Returns:

- the user service

---

addOrgName

`public UserBuilder addOrgName(String orgName)`

Add the organization name the user should be associated with.

Parameters:

- `orgName` - the organization name the user should be associated with

Returns:

- the user service

---

removeOrgName

`public UserBuilder removeOrgName(String orgName)`

Remove the organization name the user should be associated with.

Parameters:

- `orgName` - the organization name the user should not be associated with

Returns:

- the user service

---

setRoleNames

`public UserBuilder setRoleNames(Set<String> roleNames)`

Set the role names the user should be assigned to.

Parameters:

- `roleNames` - the role names the user should be assigned to

Returns:

- the user service

---

addRoleName

`public UserBuilder addRoleName(String roleName)`

Add the role name the user should be assigned to.

Parameters:

- `roleName` - the role name the user should be assigned to

Returns:

- the user service

---

removeRoleName

`public UserBuilder removeRoleName(String roleName)`

Remove the role name the user should be assigned to.

Parameters:

- `roleName` - the role name the user should not be assigned to

Returns:

- the user service

---

setSpaceNames

`public UserBuilder setSpaceNames(Set<String> spaceNames)`

Set the form space names the user should be assigned to.

Parameters:

- `spaceNames` - the form space names the user should be assigned to

Returns:

- the user service

---

addSpaceName

`public UserBuilder addSpaceName(String spaceName)`

Add the form space names the user should be assigned to.

Parameters:

- `spaceName` - the form space name the user should be assigned to

Returns:

- the user service

---

removeSpaceName

`public UserBuilder removeSpaceName(String spaceName)`

Remove the form space names the user should be assigned to.

Parameters:

- `spaceName` - the form space name the user should not be assigned to

Returns:

- the user service

---

create

```
public User create()
```

Create a new User account based on the specified parameter and return the new User value object.

Returns:

- the newly created User value object

---

update

```
public User update()
```

Update the user account specified by the loginName parameter and return the update User value object. This method will throw a IllegalArgumentException if the specified user specified by loginName parameter was not found.

Returns:

- the update User value object

---

createOrUpdate

```
public User createOrUpdate()
```

Create or update the user account specified by the given parameters, and return current User value object. If the user specified by the loginName is not found then a new user account will be created.

Returns:

- create or update the user account specified by the given parameters, and return current User value object

com.avoka.tm.test

Classes in Package com.avoka.tm.test

- AbstractJUnitTest
- JUnitTestRunner
- JUnitTestRunner.TestWrapper
- MockRegister
- MockRegistry
- MockRequest
- MockVoBuilder

Exceptions

- JUnitTestException

AbstractJUnitTest

Package:

- com.avoka.tm.test

Class AbstractJUnitTest

- java.lang.Object
- com.avoka.tm.test.AbstractJUnitTest

```
public abstract class AbstractJUnitTest
extends Object
```

Provides an abstraction utility for unit testing.

Please find the junit examples below.

JUnit Example (Multiple Tests)

This Groovy example shows how to implement a number of unit tests in a class.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import org.junit.*

class MyServiceUnitTest extends AbstractJUnitTest {

     MockRequest request
     Txn txn

     @BeforeClass
     public static void setUp() {
     }

     @Before
     public void beforeTest() {
         request = new MockRequest()
         txn = new MockVoBuilder().createTxnOpened()
     }

     @Test
     public void testSvcDefNotNull() {
         assert svcDef != null
     }

     @Test
     public void testInvokeSvc_noRequestParams() {

         Map params = [
           "svcDef": svcDef,
           "txn": txn,
           "request": request,
           "user": null
         ]

         String result = (String) new ServiceInvoker(svcDef).invoke(params)

         logger.info result

         assert result != null
     }

     @Test
     public void testInvokeSvc_branch3RequestParam() {

         request.setParameter("branch3", "test")

         Map params = [
           "svcDef": svcDef,
           "txn": txn,
```

```
            "request": request,
            "user": null
        ]

        String result = (String) new ServiceInvoker(svcDef).invoke(params)

        logger.info result

        assert "branch3".equals(result)
    }

    @After
    public void afterTest() {
        txn = null
        request = null
    }

    @AfterClass
    public static void tearDown() {
    }
}
```

Since:

- 5.1.4

Field Summary

Fields

Modifier and Type	Field and Description
SvcDef	svcDef Service definition.
Map	testParams Test parameters.

Constructor Summary

Constructors

Constructor and Description
AbstractJUnitTest()

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
void	invoke(SvcDef svcDef, Map testParams) Invoke all unit test methods (annotated with org.junit.Test) in the class.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

testParams

public Map testParams

Test parameters.

svcDef

public SvcDef svcDef

Service definition.

Constructor Detail

AbstractJUnitTest

```
public AbstractJUnitTest()
```

Method Detail

invoke

```
public void invoke(SvcDef svcDef,
                   Map testParams)
```

Invoke all unit test methods (annotated with `org.junit.Test`) in the class.

Parameters:

- `svcDef` - service definition
- `testParams` - test parameters

JUnitTestException

Package:

- com.avoka.tm.test

Class JUnitTestException

- java.lang.Object
- java.lang.Throwable
- java.lang.Exception
- java.lang.RuntimeException
- com.avoka.tm.test.JUnitTestException

All Implemented Interfaces:

- Serializable

```
public class JUnitTestException
extends RuntimeException
```

Provides a runtime exception for unit testing.

Since:

- 5.1.4

See Also:

- Serialized Form

Constructor Summary

Constructors

Constructor and Description
`JUnitTestException(Result result, String xmlReport)` Construct junit runtime exception based on result.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`Result`	`getResult()` Get JUnit result object.
`String`	`getXmlReport()` Get xml report.
`String`	`toString()` Return a string representation of this object.

Methods inherited from class java.lang.Throwable

`addSuppressed`, `fillInStackTrace`, `getCause`, `getLocalizedMessage`, `getMessage`, `getStackTrace`, `getSuppressed`, `initCause`, `printStackTrace`, `printStackTrace`, `printStackTrace`, `setStackTrace`

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Constructor Detail

JUnitTestException

```
public JUnitTestException(Result result,
                          String xmlReport)
```

Construct junit runtime exception based on result.

Parameters:

- `result` - junit result object
- `xmlReport` - junit xml report

Method Detail

getResult

`public Result getResult()`

Get JUnit result object.

Returns:

- JUnit result object

---

getXmlReport

`public String getXmlReport()`

Get xml report.

Returns:

- xml report

---

toString

`public String toString()`

Return a string representation of this object.

Overrides:

- `toString` in class `Throwable`

Returns:

- a string representation of this object.

JUnitTestRunner

Package:

- com.avoka.tm.test

Class JUnitTestRunner

- java.lang.Object
- com.avoka.tm.test.JUnitTestRunner

```
public class JUnitTestRunner
extends Object
```

Provides a runner utility for unit testing.

Please find the JUnit examples below.

Run Multiple Unit Tests Example

This Groovy example shows how to run a number of unit tests in a class with JUnitTestRunner.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import org.junit.*
import static org.junit.Assert.*

class MyUnitTest {

     public void invoke(SvcDef svcDef, Map testParams) {
         JUnitTestRunner.runTests(JUnitTest.getClass())
     }
}

public class JUnitTest {

    Object myObject

    @Test
    public void someTestCalculation() {
        assertTrue(4, 2 + 2)
    }

    @Test
    public void anotherTest() {
        assertNotNull(myObject);
    }

    @Before
    public void prepareEachTest() {
        myObject = new Object()
    }

    @After
    public void afterEachTest() {
        myObject = null
    }

    @BeforeClass
    public static void setUp() {
        //prepare some global env
    }

    @AfterClass
    public static void tearDown() {
        //clean some global env
    }
}
```

Since:

- 5.1.4

Nested Class Summary

Nested Classes

Modifier and Type	Class and Description
`static class`	`JUnitTestRunner.TestWrapper` Test wrapper.

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method and Description
`static Result`	`runTests(AbstractJUnitTest junitTestInstance)` Run unit test methods (annotated with `org.junit.Test`) in junit test instance.
`static Result`	`runTests(Class<?> junitTestClass)` Run unit test methods (annotated with `org.junit.Test`) in junit test class.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Method Detail

runTests

```
public static Result runTests(AbstractJUnitTest junitTestInstance)
```

Run unit test methods (annotated with `org.junit.Test`) in junit test instance.

Parameters:

- `junitTestInstance` - junit test instance with unit test methods

Returns:

- result with collected and summarized successful information from running multiple tests

---

runTests

```
public static Result runTests(Class<?> junitTestClass)
```

Run unit test methods (annotated with `org.junit.Test`) in junit test class.

Parameters:

- `junitTestClass` - junit test class with unit test methods

Returns:

- result with collected and summarized information from running multiple tests

JUnitTestRunner.TestWrapper

Package:

- com.avoka.tm.test

Class JUnitTestRunner.TestWrapper

- java.lang.Object
- com.avoka.tm.test.JUnitTestRunner.TestWrapper

Enclosing class:

- JUnitTestRunner

```
public static class JUnitTestRunner.TestWrapper
extends Object
```

Test wrapper.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
int	countTestCases() Get test cases count.
boolean	equals(Object obj)
String	getName() Get test name.
int	hashCode()
void	run(TestResult result) Run test.
String	toString()

Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

Method Detail

---

getName

```
public String getName()
```

Get test name.

Returns:

- test name

---

countTestCases

```
public int countTestCases()
```

Get test cases count.

Returns:

- test cases count

---

run

```
public void run(TestResult result)
```

Run test.

Parameters:

- `result` - test result

See Also:

- `junit.framework.Test#run(junit.framework.TestResult)`

---

equals

```
public boolean equals(Object obj)
```

Overrides:

- `equals` in class `Object`

---

hashCode

```
public int hashCode()
```

Overrides:

- `hashCode` in class `Object`

---

toString

```
public String toString()
```

Overrides:

- `toString` in class `Object`

MockRegister

Package:

- com.avoka.tm.test

Class MockRegister

- java.lang.Object
- com.avoka.tm.test.MockRegister

```
public class MockRegister
extends Object
```

Provides builder pattern support for use of the MockRegister.
This class can be used to make test development easier, allowing you to bypass calls to external services (via HttpRequest) or internal services (via ServiceInvoker), instead returning a predefined result.
This is useful to test a service even when other services it depends on are not available, or it is desired to test the service in isolation.

This class supports two usage patterns: new MockRegister.when(HttpRequest).thenReturn(HttpResponse) new MockRegister.when(SvcDef).thenReturn (Object)

Register HttpRequest example

This Fluent example shows how to test a service that itself makes a GetRequest call to an external service.
The code below is the JUnit test script for the service.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

class UnitTest extends AbstractJUnitTest {

    @Test
    void test1() throws Exception {
        // Register mock response for the matching request
        GetRequest request = new GetRequest('https://service.mycorp.com/secure/rest/accounts/')
          .setBasicAuth(username, password);

        HttpResponse response = new HttpResponse()
           .setStatus(401);

        new MockRegister().when(request).thenReturn(response);

        // Call service
        Map params = [
            "svcDef": svcDef,
        ]

        String result = (String) new ServiceInvoker(svcDef).invoke(params)

        // Test Results
        ...
    }
}
```

Register SvcDef example

This Fluent example shows how to test a service that itself makes call to another TM service using ServiceInvoker.
The code below is the JUnit test script for the service.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

class UnitTest extends AbstractJUnitTest {
```

```
    @Test
    void test1() throws Exception {
        // Register mock response for the matching request
        // Register the svcDef and response which will be called
        Map svcB = [
            "name": "Service B",
            "versionNumber": 1,
            "clientCode": "maguire"
        ]
        SvcDef svcDefB = new SvcDef(svcB);
        String response = "{ 'status': 'OK' }"

        new MockRegister().when(svcDefB).thenReturn(response);

        // Call service
        Map params = [
            "svcDef": svcDef
        ]

        String result = (String) new ServiceInvoker(svcDef).invoke *
        // Test Results
        ...
    }
}
```

Since:

- 5.1.7

Constructor Summary

Constructors

Constructor and Description
`MockRegister()`

Method Summary

All Methods Static Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`static void`	`clear()` Clear the Mock Register of all mock values.
`static HttpResponse`	`getHttpResponse(HttpRequest request)` Return the HttpResponse registered for the given request, if any.
`static Object`	`getServiceResult(String serviceName, Integer versionNumber, String clientCode)` Return the pair object registered for the given service, if any.
`static boolean`	`hasServiceResult(String serviceName, Integer versionNumber, String clientCode)` Check if there is a registered entry for service.
`MockRegister`	`thenReturn(HttpResponse response)` Resolves the previously set HttpRequest (via when(HttpRequest) and registers the given HttpResponse for it in the mock test request registry.
`MockRegister`	`thenReturn(Object serviceResult)` Resolves the previously set SvcDef (via when(SvcDef) and registers the given Object for it in the mock test service registry.
`MockRegister`	`when(HttpRequest request)` Set the mock test context request registry key (the HttpRequest).
`MockRegister`	`when(SvcDef svcDef)` Set the mock test context service registry key (the SvcDef).

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

MockRegister

```
public MockRegister()
```

Method Detail

when

```
public MockRegister when(HttpRequest request)
```

Set the mock test context request registry key (the HttpRequest). This value is used when thenReturn(HttpResponse) is called. This method in itself has no effect on the registry. Ensure to follow it up with thenReturn(HttpResponse).

Parameters:

- `request` - http request

Returns:

- the current MockRegister instance

---

when

```
public MockRegister when(SvcDef svcDef)
```

Set the mock test context service registry key (the SvcDef). This value is used when thenReturn(Object) is called. This method in itself has no effect on the registry. Ensure to follow it up with thenReturn(Object).

Parameters:

- `svcDef` - service definition

Returns:

- the current MockRegister instance

---

thenReturn

```
public MockRegister thenReturn(HttpResponse response)
```

Resolves the previously set HttpRequest (via when(HttpRequest) and registers the given HttpResponse for it in the mock test request registry.

Parameters:

- `response` - the HttpResponse that should be returned when a HttpRequest matching the stored request is processed.

Returns:

- the current MockRegister instance

---

thenReturn

```
public MockRegister thenReturn(Object serviceResult)
```

Resolves the previously set SvcDef (via when(SvcDef) and registers the given Object for it in the mock test service registry.

Parameters:

- `serviceResult` - the result object that should be returned when ServiceInvoker.invoke is called on a service matching the stored SvcDef.

Returns:

- the current MockRegister instance

---

getHttpResponse

```
public static HttpResponse getHttpResponse(HttpRequest request)
```

Return the HttpResponse registered for the given request, if any.

Parameters:

- `request` - the request (required)

Returns:

- the response registered for this request, or null if none was found.

---

getServiceResult

```
public static Object getServiceResult(String serviceName,
                                       Integer versionNumber,
                                       String clientCode)
```

Return the pair object registered for the given service, if any. Use hasServiceResult if you want to check `null` registered.

Parameters:

- `serviceName` - service name
- `versionNumber` - service version number
- `clientCode` - service client code

Returns:

- the service / result object registered for this service, or null if none was found.

---

hasServiceResult

```
public static boolean hasServiceResult(String serviceName,
                                       Integer versionNumber,
                                       String clientCode)
```

Check if there is a registered entry for service.

Parameters:

- `serviceName` - service name (required)
- `versionNumber` - service version number
- `clientCode` - service client code

Returns:

- true if there is, otherwise false

---

clear

```
public static void clear()
```

Clear the Mock Register of all mock values.

MockRegistry

Package:

- com.avoka.tm.test

Class MockRegistry

- java.lang.Object
- com.avoka.tm.test.MockRegistry

```
public class MockRegistry
extends Object
```

Provides thread local registry for mock HTTP request/response pairs and service/result object pairs. These are used in Fluent tests to mock up calls to external and internal services.

Since:

- 5.1.7

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method and Description
`static void`	`clear()` Clear registries.
`static Map<Object,Object>`	`getRegistry()` Get the mock object registry.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Method Detail

---

getRegistry

```
public static Map<Object,Object> getRegistry()
```

Get the mock object registry.

Returns:

- the service registry

---

clear

```
public static void clear()
```

Clear registries. This is important to avoid polluting non-test code paths.

MockRequest

Package:

- com.avoka.tm.test

Class MockRequest

- java.lang.Object
- com.avoka.tm.test.MockRequest

```
public class MockRequest
extends Object
```

Mock implementation of HttpServletRequest.

Implements all of the methods from the standard HttpServletRequest class plus helper methods to aid setting up a request.

This class was adapted from Apache Click.

Since:

- 5.0

See Also:

- `HttpServletRequest`

Field Summary

Fields

Modifier and Type	Field and Description
`static String`	`REMOTE_USER` The REMOTE_USER header.

Constructor Summary

Constructors

Constructor and Description
`MockRequest()` Create new MockRequest.
`MockRequest(Locale locale)` Create new MockRequest for the specified local.
`MockRequest(Locale locale, ServletContext servletContext)` Create a new MockRequest for the specified locale and servletContext.
`MockRequest(Locale locale, ServletContext servletContext, HttpSession session)` Create a new MockRequest for the specified arguments.
`MockRequest(Locale locale, String contextPath, String servletPath, ServletContext servletContext, HttpSession session)` Create a new MockRequest for the specified arguments.
`MockRequest(ServletContext servletContext)` Create a new MockRequest for the specified context.

Method Summary

All Methods Instance Methods Concrete Methods Deprecated Methods

Modifier and Type	Method and Description
`MockRequest`	`addCookie(Cookie cookie)` Add a new cookie.
`MockRequest`	`addHeader(String name, String value)`

	Add a header to the request.
`Object`	`getAttribute(String name)` Get an attribute.
`Enumeration<String>`	`getAttributeNames()` Get the names of all of the values.
`String`	`getAuthType()` Get the auth type.
`String`	`getCharacterEncoding()` Get the current character encoding.
`int`	`getContentLength()` Return the length of the content.
`String`	`getContentType()` If useMultiPartContentType set as true return the correct content-type.
`String`	`getContextPath()` Returns the portion of the request URI that indicates the context of the request.
`Cookie[]`	`getCookies()` Get all of the cookies for this request.
`long`	`getDateHeader(String name)` Get the given header as a date.
`String`	`getForward()` Returns the url that was forwarded to, otherwise return null.
`String`	`getHeader(String name)` Get the given header value.
`Enumeration<String>`	`getHeaderNames()` Get the names of all of the headers.
`Map<String,List<String>>`	`getHeaders()` Return the map of headers for this request.
`Enumeration<String>`	`getHeaders(String name)` Get enumeration of all header values with the given name.
`List<String>`	`getIncludes()` Returns the list of server side included url's.
`ServletInputStream`	`getInputStream()` Returns an input stream if there has been added some uploaded files.
`int`	`getIntHeader(String name)` Get the given header as an int.
`String`	`getLocalAddr()` Return the local address, *"127.0.0.1"*.
`Locale`	`getLocale()` Get the locale of the request.
`Enumeration<Locale>`	`getLocales()` Return all the accepted locales.
`String`	`getLocalName()` Return the local name, *"127.0.0.1"*.
`int`	`getLocalPort()` Return the local port, *80*.
`String`	`getMethod()` Get the method.
`String`	`getParameter(String name)` Get the request parameter with the given name.
`Map<String,String[]>`	`getParameterMap()`

	Get the map of all of the parameters.
`Enumeration<String>`	`getParameterNames()` Get the names of all of the parameters.
`String[]`	`getParameterValues(String name)` Get the values for the given parameter.
`String`	`getPathInfo()` Get the path info.
`String`	`getPathTranslated()` Always returns null.
`String`	`getProtocol()` Get the protocol.
`String`	`getQueryString()` Get the query string part of the request.
`BufferedReader`	`getReader()` This feature is not implemented at this time as we are not supporting binary servlet input.
`String`	`getRealPath(String name)` Deprecated. Use ServletContext.getRealPath(String) instead.
`String`	`getRemoteAddr()` Get the remote address of the client.
`String`	`getRemoteHost()` Get the remote host.
`int`	`getRemotePort()` Return the remote port, *80*.
`String`	`getRemoteUser()` Return the name of the userPrincipal if set, otherwise the value of the "REMOTE_USER" header.
`RequestDispatcher`	`getRequestDispatcher(String path)` Returns a RequestDispatcher for the specified path.
`String`	`getRequestedSessionId()` Get the requested session id.
`String`	`getRequestURI()` Returns context path and servlet path concatenated, typically /applicationClassName /applicationClassName.
`StringBuffer`	`getRequestURL()` Returns (an attempt at) a reconstructed URL based on it's constituent parts.
`String`	`getScheme()` Get the scheme http, https, or ftp.
`String`	`getServerName()` Get the host server name to which the request was sent.
`int`	`getServerPort()` Returns the port number to which the request was sent.
`ServletContext`	`getServletContext()` Return the servlet context.
`String`	`getServletPath()` Return a String containing the name or path of the servlet being called.
`HttpSession`	`getSession()` Returns the current HttpSession associated with this request.
`HttpSession`	`getSession(boolean create)` Returns the current HttpSession associated with this request.
`Principal`	`getUserPrincipal()` Get the user principal.

void	initialize() Reset the request back to a default state.
boolean	isPost() Return whether the request is a post or not.
boolean	isRequestedSessionIdFromCookie() Check whether session id is from a cookie.
boolean	isRequestedSessionIdFromUrl() Check whether session id is from a url rewrite.
boolean	isRequestedSessionIdFromURL() Check whether session id is from a url rewrite.
boolean	isRequestedSessionIdValid() Check whether the session id is valid.
boolean	isSecure() Always returns false.
boolean	isUserInRole(String role) Returns true if the authenticated user is included in the given role, false otherwise.
void	removeAttribute(String name) Remove the given attribute.
MockRequest	removeParameter(String name) Remove the specified parameter.
void	reset() Delegate to initialize method.
void	setAttribute(String name, Object o) Set the given attribute.
void	setAuthType(String authType) Set the auth type.
void	setCharacterEncoding(String encoding) Set the character encoding.
MockRequest	setContentType(String contentType) Set the request content type.
MockRequest	setContextPath(String contextPath) Set the portion of the request URI that indicates the context of the request.
void	setCookies(Cookie[] theCookies) Set the cookies.
MockRequest	setHeader(String name, String... values) Set request header values.
MockRequest	setHeader(String name, String value) Set request header value.
MockRequest	setHttpSession(HttpSession session) Set the request's session instance.
MockRequest	setMethod(String method) Set the method.
MockRequest	setParameter(String name, String value) Set a parameter.
MockRequest	setParameter(String name, String[] values) Set the specified parameter name to the array of strings.
MockRequest	setParameters(Map<String,Object> parameters) Sets a map of parameters.
MockRequest	setPathInfo(String path) Set the path that this request is supposed to be serving.
void	setRequestURL(String requestURL)

	Provide a convenience method for setting the request URL.
MockRequest	setScheme(String scheme) Set the request's scheme, for example http, https, or ftp.
MockRequest	setServerName(String serverName) Sets the host server name to which the request was sent.
MockRequest	setServerPort(int serverPort) Set the port number to which the request was sent.
MockRequest	setServletContext(ServletContext servletContext) Set the request's servletContext instance.
MockRequest	setServletPath(String servletPath) Set the string containing the name or path of the servlet being called.
MockRequest	setSession(HttpSession session) Set the current HttpSession associated with this request.
MockRequest	setUseMultiPartContentType(boolean useMultiPartContentType) True will force Request to generate multiPart ContentType and ContentLength.
MockRequest	setUserPrincipal(Principal userPrincipal) Set the user principal.
String	toString() Returns the String representation of the mock request.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

REMOTE_USER

```
public static final String REMOTE_USER
```

The REMOTE_USER header.

See Also:

- Constant Field Values

Constructor Detail

MockRequest

```
public MockRequest()
```

Create new MockRequest.

MockRequest

```
public MockRequest(Locale locale)
```

Create new MockRequest for the specified local.

Parameters:

- `locale` - locale for this request

MockRequest

```
public MockRequest(ServletContext servletContext)
```

Create a new MockRequest for the specified context.

Parameters:

- `servletContext` - the servletContext for this request

MockRequest

```
public MockRequest(Locale locale,
                   ServletContext servletContext)
```

Create a new MockRequest for the specified locale and servletContext.

Parameters:

- `locale` - locale for this request
- `servletContext` - the servletContext for this request

MockRequest

```
public MockRequest(Locale locale,
                   ServletContext servletContext,
                   HttpSession session)
```

Create a new MockRequest for the specified arguments.

Parameters:

- `locale` - The request locale, or null to use the default locale
- `session` - The session object
- `servletContext` - The current servlet context

MockRequest

```
public MockRequest(Locale locale,
                   String contextPath,
                   String servletPath,
                   ServletContext servletContext,
                   HttpSession session)
```

Create a new MockRequest for the specified arguments.

Parameters:

- `locale` - The request locale, or null to use the default locale
- `contextPath` - the request context path
- `servletPath` - the request servlet path
- `servletContext` - The current servlet context
- `session` - the request session

Method Detail

---

setServletContext

```
public MockRequest setServletContext(ServletContext servletContext)
```

Set the request's servletContext instance.

Parameters:

- `servletContext` - the new ServletContext instance

Returns:

- the MockRequest object

---

setHttpSession

```
public MockRequest setHttpSession(HttpSession session)
```

Set the request's session instance.

Parameters:

- `session` - the new HttpSession instance

Returns:

- the MockRequest object

---

addCookie

```
public MockRequest addCookie(Cookie cookie)
```

Add a new cookie.

Parameters:

- `cookie` - The cookie

Returns:

- the MockRequest object

---

addHeader

```
public MockRequest addHeader(String name,
                             String value)
```

Add a header to the request.

Parameters:

- `name` - the name of the header to add
- `value` - the value

Returns:

- the MockRequest object

---

setHeader

```
public MockRequest setHeader(String name,
                             String value)
```

Set request header value. The existing header value will be replaced.

Parameters:

- `name` - the name of the header to set
- `value` - the header value

Returns:

- the MockRequest object

---

setHeader

```
public MockRequest setHeader(String name,
                             String... values)
```

Set request header values. The existing header values will be replaced.

Parameters:

- `name` - the name of the header to set
- `values` - the header values

Returns:

- the MockRequest object

---

getAttribute

```
public Object getAttribute(String name)
```

Get an attribute.

Parameters:

- `name` - The attribute name

Returns:

- The value, or null

---

getAttributeNames

```
public Enumeration<String> getAttributeNames()
```

Get the names of all of the values.

Returns:

- The names

---

getAuthType

```
public String getAuthType()
```

Get the auth type.

Returns:

- The auth type

---

getCharacterEncoding

```
public String getCharacterEncoding()
```

Get the current character encoding.

Returns:

- The character encoding

---

setUseMultiPartContentType

```
public MockRequest setUseMultiPartContentType(boolean useMultiPartContentType)
```

True will force Request to generate multiPart ContentType and ContentLength.

Parameters:

- `useMultiPartContentType` - true if the request is multi-part, false otherwise

Returns:

- the MockRequest object

---

getContentLength

```
public int getContentLength()
```

Return the length of the content. This is always -1 except if useMultiPartContentType set as true. Then the length will be the length of the generated request.

Returns:

- -1 if useMultiPartContentType is false. Else the length of the generated request.

---

setContentType

```
public MockRequest setContentType(String contentType)
```

Set the request content type.

Parameters:

- `contentType` - the request content type.

Returns:

- the MockRequest object

---

getContentType

```
public String getContentType()
```

If useMultiPartContentType set as true return the correct content-type. If the contentType property is defined then this value will be returned.

Returns:

- The correct multipart content-type if useMultiPartContentType is true. Else null.

---

getCookies

```
public Cookie[] getCookies()
```

Get all of the cookies for this request.

Returns:

- The cookies

---

getDateHeader

```
public long getDateHeader(String name)
```

Get the given header as a date.

Parameters:

- `name` - The header name

Returns:

- The date, or -1 if header not found

---

getHeader

```
public String getHeader(String name)
```

Get the given header value.

Parameters:

- `name` - The header name

Returns:

- The header value or null

---

getHeaderNames

```
public Enumeration<String> getHeaderNames()
```

Get the names of all of the headers.

Returns:

- The header names

---

getHeaders

```
public Enumeration<String> getHeaders(String name)
```

Get enumeration of all header values with the given name.

Parameters:

- `name` - The name

Returns:

- The header values

---

getHeaders

```
public Map<String,List<String>> getHeaders()
```

Return the map of headers for this request.

Returns:

- the map of headers for this request

---

getInputStream

```
public ServletInputStream getInputStream()
                                 throws IOException
```

Returns an input stream if there has been added some uploaded files. Use `#addFile(String, File, String)` to add some uploaded files.

Returns:

- The input stream

Throws:

- `IOException` - If an I/O related problem occurs

---

getIntHeader

```
public int getIntHeader(String name)
```

Get the given header as an int.

Parameters:

- `name` - The header name

Returns:

- The header value or -1 if header not found

---

getLocale

```
public Locale getLocale()
```

Get the locale of the request. Attempts to decode the Accept-Language header and if not found returns the default locale of the JVM.

Returns:

- The locale

---

getLocales

```
public Enumeration<Locale> getLocales()
```

Return all the accepted locales. This implementation always returns just one.

Returns:

- The locales

---

getMethod

```
public String getMethod()
```

Get the method. The returned string will be in upper case eg. POST.

Returns:

- The method

---

getParameter

```
public String getParameter(String name)
```

Get the request parameter with the given name.

Parameters:

- `name` - The parameter name

Returns:

- The parameter value, or null

---

getParameterMap

```
public Map<String,String[]> getParameterMap()
```

Get the map of all of the parameters.

Returns:

- The parameters

---

getParameterNames

```
public Enumeration<String> getParameterNames()
```

Get the names of all of the parameters.

Returns:

- The parameter names

---

getParameterValues

```
public String[] getParameterValues(String name)
```

Get the values for the given parameter.

Parameters:

- `name` - The name of the parameter

Returns:

- The return values

---

getPathInfo

```
public String getPathInfo()
```

Get the path info.

Returns:

- The path info

---

getPathTranslated

```
public String getPathTranslated()
```

Always returns null.

Returns:

- null

---

getProtocol

```
public String getProtocol()
```

Get the protocol.

Returns:

- Always HTTP/1.1

---

getQueryString

```
public String getQueryString()
```

Get the query string part of the request.

Returns:

- The query string

---

getReader

```
public BufferedReader getReader()
                      throws IOException
```

This feature is not implemented at this time as we are not supporting binary servlet input. This functionality may be added in the future.

Returns:

- The reader

Throws:

- `IOException` - If an I/O related problem occurs

---

getRealPath

```
public String getRealPath(String name)
```

Deprecated. Use ServletContext.getRealPath(String) instead.

Deprecated method - should not be used.

Parameters:

- `name` - The name

Returns:

- The path

---

getRemoteAddr

```
public String getRemoteAddr()
```

Get the remote address of the client.

Returns:

- Always 127.0.0.1

---

getRemoteHost

```
public String getRemoteHost()
```

Get the remote host.

Returns:

- Always localhost

---

getRemoteUser

```
public String getRemoteUser()
```

Return the name of the userPrincipal if set, otherwise the value of the "REMOTE_USER" header.

To set the remote user, create an instance of a `MockPrincipal` and set it on the request through the method `setUserPrincipal(java.security.Principal)`.

Returns:

- the name of the remote user

---

getLocalAddr

```
public String getLocalAddr()
```

Return the local address, *"127.0.0.1"*.

Returns:

- "127.0.0.1" as the local address

---

getLocalName

```
public String getLocalName()
```

Return the local name, *"127.0.0.1"*.

Returns:

- "127.0.0.1" as the local name

---

getLocalPort

```
public int getLocalPort()
```

Return the local port, *80*.

Returns:

- 80 as the local port

---

getRemotePort

```
public int getRemotePort()
```

Return the remote port, *80*.

Returns:

- 80 as the remote port

---

getRequestDispatcher

```
public RequestDispatcher getRequestDispatcher(String path)
```

Returns a RequestDispatcher for the specified path. The dispatcher will not dispatch to the resource. It only records the specified path so that one can test if the correct path was dispatched to.

Parameters:

- `path` - a String specifying the pathname to the resource

Returns:

- a dispatcher for the specified path

---

getRequestedSessionId

```
public String getRequestedSessionId()
```

Get the requested session id. Always returns the id of the current session.

Returns:

- The session id

---

getRequestURI

```
public String getRequestURI()
```

Returns context path and servlet path concatenated, typically /applicationClassName/applicationClassName.

Returns:

- The path value

See Also:

- `javax.servlet.http.HttpServletRequest#getRequestURI()`

---

setRequestURL

```
public void setRequestURL(String requestURL)
```

Provide a convenience method for setting the request URL.

Parameters:

- `requestURL` - the request URL to set

---

getRequestURL

```
public StringBuffer getRequestURL()
```

Returns (an attempt at) a reconstructed URL based on it's constituent parts. If the requestURL property is set this value will be returned instead.

Returns:

- a StringBuffer object containing the reconstructed URL

See Also:

- `javax.servlet.http.HttpServletRequest#getRequestURL()`

---

isPost

```
public boolean isPost()
```

Return whether the request is a post or not.

Returns:

- true if the request is a post, false otherwise

---

getScheme

`public String getScheme()`

Get the scheme http, https, or ftp.

Returns:

- the scheme used by this request

---

setScheme

`public MockRequest setScheme(String scheme)`

Set the request's scheme, for example http, https, or ftp.

Parameters:

- `scheme` - the request's scheme

Returns:

- the MockRequest object

---

getServerName

`public String getServerName()`

Get the host server name to which the request was sent.

Returns:

- always the host server name

---

setServerName

`public MockRequest setServerName(String serverName)`

Sets the host server name to which the request was sent.

Parameters:

- `serverName` - the server name the request was sent to

Returns:

- the MockRequest object

---

getServerPort

`public int getServerPort()`

Returns the port number to which the request was sent.

Returns:

- the server port to which the request was sent

---

setServerPort

`public MockRequest setServerPort(int serverPort)`

Set the port number to which the request was sent.

Parameters:

- `serverPort` - the port number to which the request was sent

Returns:

- the MockRequest object

---

getContextPath

`public String getContextPath()`

Returns the portion of the request URI that indicates the context of the request.

Returns:

- the portion of the request URI that indicates the context of the request.

---

setContextPath

```
public MockRequest setContextPath(String contextPath)
```

Set the portion of the request URI that indicates the context of the request.

Parameters:

- `contextPath` - the portion of the request URI that indicates the context of the request.

Returns:

- the MockRequest object

---

getServletContext

```
public ServletContext getServletContext()
```

Return the servlet context.

Returns:

- the servlet context.

Since:

- Servlet 3.0

---

getServletPath

```
public String getServletPath()
```

Return a String containing the name or path of the servlet being called.

Returns:

- The servlet path

---

setServletPath

```
public MockRequest setServletPath(String servletPath)
```

Set the string containing the name or path of the servlet being called.

Parameters:

- `servletPath` - a String containing the name or path of the servlet being called

Returns:

- the MockRequest object

---

getSession

```
public HttpSession getSession()
```

Returns the current HttpSession associated with this request.

Returns:

- the session associated with this request

---

setSession

```
public MockRequest setSession(HttpSession session)
```

Set the current HttpSession associated with this request.

Parameters:

- `session` - the HttpSession to associate with this request

Returns:

- the MockRequest object

---

getSession

```
public HttpSession getSession(boolean create)
```

Returns the current HttpSession associated with this request.

Parameters:

- `create` - if true creates a new session if one does not exist

Returns:

- the current HttpSession associated with this request.

---

getUserPrincipal

```
public Principal getUserPrincipal()
```

Get the user principal. If no user principal was set this method will create a user principal for the `getRemoteUser()`.

Returns:

- the user principal

---

setUserPrincipal

```
public MockRequest setUserPrincipal(Principal userPrincipal)
```

Set the user principal.

Parameters:

- `userPrincipal` - the user principal

Returns:

- the MockRequest object

---

initialize

```
public final void initialize()
```

Reset the request back to a default state.

---

reset

```
public void reset()
```

Delegate to initialize method.

---

isRequestedSessionIdFromCookie

```
public boolean isRequestedSessionIdFromCookie()
```

Check whether session id is from a cookie. Always returns true.

Returns:

- Always true

---

isRequestedSessionIdFromUrl

```
public boolean isRequestedSessionIdFromUrl()
```

Check whether session id is from a url rewrite. Always returns false.

Returns:

- Always false

---

isRequestedSessionIdFromURL

```
public boolean isRequestedSessionIdFromURL()
```

Check whether session id is from a url rewrite. Always returns false.

Returns:

- Always false

---

isRequestedSessionIdValid

```
public boolean isRequestedSessionIdValid()
```

Check whether the session id is valid.

Returns:

- Always true

---

isSecure

```
public boolean isSecure()
```

Always returns false.

Returns:

- Always false

---

isUserInRole

```
public boolean isUserInRole(String role)
```

Returns true if the authenticated user is included in the given role, false otherwise.

To mock up roles for a user, create a `user principal` and set the necessary roles. See `MockPrincipal` for an example.

Parameters:

- `role` - the role name

Returns:

- true if the user is included in the specified role, false otherwise

---

removeAttribute

```
public void removeAttribute(String name)
```

Remove the given attribute.

Parameters:

- `name` - The name of the attribute

---

setAttribute

```
public void setAttribute(String name,
                         Object o)
```

Set the given attribute.

Parameters:

- `name` - The attribute name
- `o` - The value to set

---

setAuthType

```
public void setAuthType(String authType)
```

Set the auth type.

Parameters:

- `authType` - The auth type

---

setCharacterEncoding

```
public void setCharacterEncoding(String encoding)
                          throws UnsupportedEncodingException
```

Set the character encoding.

Parameters:

- `encoding` - The character encoding

Throws:

- `UnsupportedEncodingException` - If encoding not supported

---

setCookies

```
public void setCookies(Cookie[] theCookies)
```

Set the cookies.

Parameters:

- `theCookies` - The cookies

---

setMethod

```
public MockRequest setMethod(String method)
```

Set the method.

Parameters:

- `method` - The method

Returns:

- the MockRequest object

---

setParameter

```
public MockRequest setParameter(String name,
                                String value)
```

Set a parameter.

Parameters:

- `name` - The name
- `value` - The value

Returns:

- the MockRequest object

---

setParameter

```
public MockRequest setParameter(String name,
                                String[] values)
```

Set the specified parameter name to the array of strings.

Parameters:

- `name` - name of the parameter
- `values` - the parameter values

Returns:

- the MockRequest object

---

removeParameter

```
public MockRequest removeParameter(String name)
```

Remove the specified parameter.

Parameters:

- `name` - the parameter name to remove

Returns:

- the MockRequest object

---

setParameters

```
public MockRequest setParameters(Map<String,Object> parameters)
```

Sets a map of parameters.

Parameters:

- `parameters` - the parameters to set

Returns:

- the MockRequest object

---

setPathInfo

```
public MockRequest setPathInfo(String path)
```

Set the path that this request is supposed to be serving. The path is relative to the web application root and should start with a / character

Parameters:

- `path` - specifies the request path to serve

Returns:

- the MockRequest object

---

getForward

```
public String getForward()
```

Returns the url that was forwarded to, otherwise return null.

Returns:

- url that was forwarded to

See Also:

- `servlet.MockRequestDispatcher`

---

getIncludes

```
public List<String> getIncludes()
```

Returns the list of server side included url's.

Returns:

- list of urls that were included

See Also:

- `servlet.MockRequestDispatcher`

---

toString

```
public String toString()
```

Returns the String representation of the mock request.

Overrides:

- `toString` in class `Object`

Returns:

- string representation of the mock request

MockVoBuilder

Package:

- com.avoka.tm.test

Class MockVoBuilder

- java.lang.Object
- com.avoka.tm.test.MockVoBuilder

```
public class MockVoBuilder
extends Object
```

Provides a mock entity value object creation service for unit testing.

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`MockVoBuilder()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`Form`	`createForm()` Create a test form entity with a random name.
`Job`	`createJobInProgress()` Create an Job with a status of 'In Progress'.
`Job`	`createJobInProgressWithXml(String formXmlData)` Create an Job with a status of 'In Progress', and a start submission with the given form XML data.
`Txn`	`createTxnCompletedWithFormAndXml(Form form)` Create a completed transaction entity with the given form.
`Txn`	`createTxnCompletedWithFormAndXml(Form form, String formXmlData)` Create a completed transaction entity with the given form and XML form data.
`Txn`	`createTxnCompletedWithXml(String formXmlData)` Create a completed transaction entity with the given XML form data.
`Txn`	`createTxnDeliveryReadyWithXml(String formXmlData)` Create a completed transaction entity with the given form and XML form data.
`Txn`	`createTxnOpened()` Create a opened transaction entity.
`Txn`	`createTxnSavedWithXml(String formXmlData)` Create a saved transaction entity with the given XML form data.
`Txn`	`createTxnSubmittedWithFormAndXml(Form form)` Create a submitted transaction entity with the given form.
`Txn`	`createTxnSubmittedWithFormAndXml(Form form, String formXmlData)` Create a submitted transaction entity with the given form and XML form data.
`Txn`	`createTxnSubmittedWithXml(String formXmlData)` Create a submitted transaction entity with the given XML form data.
`Txn`	`createTxnTaskWithXml(String formXmlData)` Create an assigned Submission form task entity with the given XML form data.
`User`	`createUserLocal()`

	Create a local type user with a random login name.
User	createUserSso() Create a SSO type user with a random login name.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

MockVoBuilder

```
public MockVoBuilder()
```

Method Detail

createTxnOpened

```
public Txn createTxnOpened()
```

Create a opened transaction entity. This is useful for mocking up Dynamic Data calls. The transaction record will have the status values:

```
Form Status:     Opened
Receipt Status:  null
Delivery Status: Not Ready
```

Returns:

- a opened transaction value object

---

createTxnSavedWithXml

```
public Txn createTxnSavedWithXml(String formXmlData)
```

Create a saved transaction entity with the given XML form data. The transaction record will have the status values:

```
Form Status:     Saved
Receipt Status:  null
Delivery Status: Not Ready
```

Parameters:

- `formXmlData` - the form XML data for the transaction

Returns:

- a saved transaction entity with the given XML form data

---

createTxnTaskWithXml

```
public Txn createTxnTaskWithXml(String formXmlData)
```

Create an assigned Submission form task entity with the given XML form data. The transaction record will have the status values:

```
Form Status:     Assigned
Task Type:       Form
Receipt Status:  null
Delivery Status: Not Ready
```

Parameters:

- `formXmlData` - the form XML data for the transaction

Returns:

- a assigned task transaction value object

---

createTxnSubmittedWithXml

```
public Txn createTxnSubmittedWithXml(String formXmlData)
```

Create a submitted transaction entity with the given XML form data. The transaction record will have the status values:

```
Form Status:     Submitted
Receipt Status:  null
```

```
Delivery Status: Not Ready
```

Parameters:

- `formXmlData` - the form XML data for the transaction

Returns:

- a completed transaction value object

---

createTxnSubmittedWithFormAndXml

```
public Txn createTxnSubmittedWithFormAndXml(Form form,
                                            String formXmlData)
```

Create a submitted transaction entity with the given form and XML form data. The transaction record will have the status values:

```
Form Status:     Completed
Receipt Status:  null
Delivery Status: Not Ready
```

If the form has form data extract mappings configured, then submission data extracts will be created against the transaction.

Parameters:

- `form` - the transaction form (required)
- `formXmlData` - the form XML data for the transaction (required)

Returns:

- a completed transaction value object

Since:

- 5.0.0

---

createTxnSubmittedWithFormAndXml

```
public Txn createTxnSubmittedWithFormAndXml(Form form)
```

Create a submitted transaction entity with the given form. The transaction record will have the status values:

```
Form Status:     Completed
Receipt Status:  null
Delivery Status: Not Ready
```

If the form has form data extract mappings configured, then submission data extracts will be created against the transaction.

Parameters:

- `form` - the transaction form (required)

Returns:

- a completed transaction value object

Since:

- 5.1.0

---

createTxnCompletedWithXml

```
public Txn createTxnCompletedWithXml(String formXmlData)
```

Create a completed transaction entity with the given XML form data. The transaction record will have the status values:

```
Form Status:     Completed
Receipt Status:  Ready
Delivery Status: Not Ready
```

Parameters:

- `formXmlData` - the form XML data for the transaction

Returns:

- a completed transaction value object

---

createTxnCompletedWithFormAndXml

```
public Txn createTxnCompletedWithFormAndXml(Form form,
                                            String formXmlData)
```

Create a completed transaction entity with the given form and XML form data. The transaction record will have the status values:

```
 Form Status:     Completed
 Receipt Status:  null
 Delivery Status: Not Ready
```

If the form has form data extract mappings configured, then submission data extracts will be created against the transaction.

Parameters:

- `form` - the transaction form (required)
- `formXmlData` - the form XML data for the transaction (required)

Returns:

- a completed transaction value object

---

createTxnCompletedWithFormAndXml

```
public Txn createTxnCompletedWithFormAndXml(Form form)
```

Create a completed transaction entity with the given form. The transaction record will have the status values:

```
 Form Status:     Completed
 Receipt Status:  null
 Delivery Status: Not Ready
```

If the form has form data extract mappings configured, then submission data extracts will be created against the transaction.

Parameters:

- `form` - the transaction form (required)

Returns:

- a completed transaction value object

Since:

- 5.1.0

---

createTxnDeliveryReadyWithXml

```
public Txn createTxnDeliveryReadyWithXml(String formXmlData)
```

Create a completed transaction entity with the given form and XML form data. The transaction record will have the status values:

```
 Form Status:     Completed
 Receipt Status:  Completed
 Delivery Status: Ready
```

Parameters:

- `formXmlData` - the form XML data for the transaction

Returns:

- a completed transaction value object

---

createJobInProgress

```
public Job createJobInProgress()
```

Create an Job with a status of 'In Progress'.

Returns:

- a new Job with a status of 'In Progress'

---

createJobInProgressWithXml

```
public Job createJobInProgressWithXml(String formXmlData)
```

Create an Job with a status of 'In Progress', and a start submission with the given form XML data.

Parameters:

- `formXmlData` - the form XML data for the transaction (required)

Returns:

- a new Job with a status of 'In Progress'

---

createForm

```
public Form createForm()
```

Create a test form entity with a random name.

Returns:

- test form entity with a random name.

---

createUserLocal

```
public User createUserLocal()
```

Create a local type user with a random login name.

Returns:

- local type user with a random login name

---

createUserSso

```
public User createUserSso()
```

Create a SSO type user with a random login name.

Returns:

- SSO type user with a random login name

com.avoka.tm.util

Classes in Package com.avoka.tm.util

- Contract
- DeliveryResult
- DeliveryResultBuilder
- MemCache
- Path
- Security
- Threads
- TxnUrlBuilder
- VelTemplate
- XmlDoc

Exceptions

- RedirectException

Contract

Package:

- com.avoka.tm.util

Class Contract

- java.lang.Object
- com.avoka.tm.util.Contract

```
public class Contract
extends Object
```

Provides design by contract programming validation functions.

Since:

- 5.0.0

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method and Description
`static void`	`isTrue(boolean value, String message)` Ensure the specified condition value is true, or throw an IllegalArgumentExcpetion if false.
`static void`	`notBlank(String paramValue, String paramName)` Validate specified parameter is not null, or blank throw a IllegalArgumentException if null or blank.
`static void`	`notNull(Object paramValue, String paramName)` Validate specified parameter is not null, throw a IllegalArgumentException if null.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Method Detail

---

isTrue

```
public static void isTrue(boolean value,
                          String message)
```

Ensure the specified condition value is true, or throw an IllegalArgumentExcpetion if false.

Parameters:

- `value` - the conditional value to test
- `message` - the IllegalArgumentException message if the conditional value is false

---

notNull

```
public static void notNull(Object paramValue,
                           String paramName)
```

Validate specified parameter is not null, throw a IllegalArgumentException if null.

Parameters:

- `paramValue` - the parameter value to verify not null
- `paramName` - the parameter name

---

notBlank

```
public static void notBlank(String paramValue,
                            String paramName)
```

Validate specified parameter is not null, or blank throw a IllegalArgumentException if null or blank.

Parameters:

- `paramValue` - the parameter value to verify not null, or blank
- `paramName` - the parameter name

DeliveryResult

Package:

- com.avoka.tm.util

Class DeliveryResult

- java.lang.Object
- com.avoka.fc.core.service.DeliveryResult
- com.avoka.tm.util.DeliveryResult

```
public class DeliveryResult
extends com.avoka.fc.core.service.DeliveryResult
```

Provides a transaction delivery process result value object.

Since:

- 5.0

Nested Class Summary

Nested classes/interfaces inherited from class com.avoka.fc.core.service.DeliveryResult

`com.avoka.fc.core.service.DeliveryResult.Status`

Constructor Summary

Constructors

Constructor and Description
`DeliveryResult(com.avoka.fc.core.service.DeliveryResult.Status status)` Create a new transaction delivery process result object.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`String`	`getStatus()` Return the transaction delivery status.

Methods inherited from class com.avoka.fc.core.service.DeliveryResult

`getDataPurgeTime, getDeliveryChannel, getDeliveryStatus, getMaxDeliveryAttempts, getMessage, getNextDeliveryTime, isStatusCompleted, isStatusError, isStatusInProgress, isStatusPending, isStatusReady, isStatusUndeliverable, setDataPurgeTime, setDeliveryChannel, setMaxDeliveryAttempts, setMessage, setNextDeliveryTime, toDeliveryStatus, toString`

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

DeliveryResult

```
public DeliveryResult(com.avoka.fc.core.service.DeliveryResult.Status status)
```

Create a new transaction delivery process result object.

Parameters:

- `status` - delivery status (required)

Method Detail

---

getStatus

```
public String getStatus()
```

Return the transaction delivery status.

Returns:

- the transaction delivery status

DeliveryResultBuilder

Package:

- com.avoka.tm.util

Class DeliveryResultBuilder

- java.lang.Object
- com.avoka.tm.util.DeliveryResultBuilder

```
public class DeliveryResultBuilder
extends Object
```

Provides a transaction delivery process result DeliveryResult builder.

Using the delivery result object you specify to Transact Manager whether the delivery has been completed, or what addition delivery processing is required.

Delivery statuses include:

Not Ready - the transaction is not ready for delivery Ready - the transaction is ready for delivery processing In Progress - the transaction delivery is in progress Pending - the transaction delivery is paused waiting for an external event to continue processing Completed - the transaction delivery has been completed Error - an error occurred performing delivery, and delivery may be retried Undeliverable - the transaction is undeliverable, and no further delivery attempts will be performed

Examples

Please find the transaction delivery process result builder examples below.

Delivery Completed Example

This example shows how to build a delivery Completed result.

```
return new DeliveryResultBuilder()
    .setStatus("Completed")
    .build()
```

Delivery Error Example

This example shows how to build a delivery Error result, with the given exception as the delivery message, and set to retry in 30 minutes time for a maximum of 5 delivery attempts.

```
return new DeliveryResultBuilder()
    .setMaxDeliveryAttempts(5)
    .setMessage(exception)
    .setNextDeliveryMins(30)
    .setStatus("Error")
    .build()
```

Delivery Channel Example

This example shows how set the delivery channel to the "REST Delivery Service" and set the delivery status to "Ready".

```
return new DeliveryResultBuilder()
    .setStatus("Ready")
    .setDeliveryChannel("REST Delivery Service")
    .build()
```

Since:

- 5.0

See Also:

- DeliveryResult

Constructor Summary

Constructors

Constructor and Description
`DeliveryResultBuilder()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`DeliveryResult`	`build()` Return a new transaction delivery result object based on the specified properties.
`DeliveryResultBuilder`	`setDataPurgeTime(Date dataPurgeTime)` The scheduled transaction PII data purge time once delivery has been completed.
`DeliveryResultBuilder`	`setDeliveryChannel(String deliveryChannel)` Set the delivery channel.
`DeliveryResultBuilder`	`setMaxDeliveryAttempts(int maxDeliveryAttempts)` The maximum number of delivery attempts to perform.
`DeliveryResultBuilder`	`setMessage(Object message)` The delivery process result message.
`DeliveryResultBuilder`	`setNextDeliveryMins(int minutes)` The next time in minutes to attempt to execute the delivery process.
`DeliveryResultBuilder`	`setNextDeliveryTime(Date nextDeliveryTime)` The next time to attempt to execute the delivery process.
`DeliveryResultBuilder`	`setStatus(String status)` The delivery process result status [Not Ready | Ready | In Progress | Pending | Completed | Error | Undeliverable].

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

DeliveryResultBuilder

```
public DeliveryResultBuilder()
```

Method Detail

setStatus

```
public DeliveryResultBuilder setStatus(String status)
```

The delivery process result status [Not Ready | Ready | In Progress | Pending | Completed | Error | Undeliverable].

Parameters:

- `status` - delivery process result status (required)

Returns:

- the delivery result builder

---

setMessage

```
public DeliveryResultBuilder setMessage(Object message)
```

The delivery process result message.

Parameters:

- `message` - delivery process result message

Returns:

- the delivery result builder

setNextDeliveryTime

`public DeliveryResultBuilder setNextDeliveryTime(Date nextDeliveryTime)`

The next time to attempt to execute the delivery process.

Parameters:

- `nextDeliveryTime` - the next time to attempt to delivery process execution

Returns:

- the delivery result builder

---

setNextDeliveryMins

`public DeliveryResultBuilder setNextDeliveryMins(int minutes)`

The next time in minutes to attempt to execute the delivery process.

Parameters:

- `minutes` - the next time in minutes to attempt to delivery process execution

Returns:

- the delivery result builder

---

setMaxDeliveryAttempts

`public DeliveryResultBuilder setMaxDeliveryAttempts(int maxDeliveryAttempts)`

The maximum number of delivery attempts to perform.

Parameters:

- `maxDeliveryAttempts` - the maximum number of delivery attempts to perform

Returns:

- the delivery result builder

---

setDataPurgeTime

`public DeliveryResultBuilder setDataPurgeTime(Date dataPurgeTime)`

The scheduled transaction PII data purge time once delivery has been completed.

Parameters:

- `dataPurgeTime` - the scheduled transaction PII data purge time once delivery has been completed

Returns:

- the delivery result builder

---

setDeliveryChannel

`public DeliveryResultBuilder setDeliveryChannel(String deliveryChannel)`

Set the delivery channel.

Parameters:

- `deliveryChannel` - the delivery channel

Returns:

- the delivery result builder

Since:

- 5.0.2

---

build

`public DeliveryResult build()`

Return a new transaction delivery result object based on the specified properties.

Returns:

- a new transaction delivery result object based on the specified properties.

MemCache

Package:

- com.avoka.tm.util

Class MemCache

- java.lang.Object
- com.avoka.tm.util.MemCache

```
public class MemCache
extends Object
```

Provides an in memory cache for high frequency read only access values. Cached values are keyed on the specified cache key and the executing service's Organization security context.

When using the memory cache ensure you do not put excessively large values in the cache which would impact the application servers memory usage.

Examples

Put Value in Cache

This examples puts a value in the cache with a 12 hour timeout (720 mins).

```
import com.avoka.tm.util.*

String value = "some big and expensive data..."

new MemCache()
    .setKey("Ref Data")
    .setTimeout(720)
    .setValue(value)
```

Get Value from Cache

This examples gets a value from the cache with a 12 hour cache timeout (720 mins).

```
import com.avoka.tm.util.*

String value = new MemCache()
    .setKey("Ref Data")
    .setTimeout(720)
    .getValue()
```

Since:

- 5.0.0

See Also:

- PropertyQuery

Constructor Summary

Constructors

Constructor and Description
`MemCache()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`Object`	`getValue()` Return the memory cached object, or null if not found or has expired.
`MemCache`	`setKey(String key)` Set the mem cache key parameter.
`MemCache`	`setTimeout(int timeoutMins)`

	Set the mem cache timeout in minutes.
void	setValue(Object value) Set the value to cache in memory with the specified key and cacheTimeout properties.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

MemCache

```
public MemCache()
```

Method Detail

setKey

```
public MemCache setKey(String key)
```

Set the mem cache key parameter.

Parameters:

- key - the cache key

Returns:

- the mem cache

---

setTimeout

```
public MemCache setTimeout(int timeoutMins)
```

Set the mem cache timeout in minutes.

Parameters:

- timeoutMins - the mem cache timeout in minutes, must be a positive value.

Returns:

- the mem cache

---

getValue

```
public Object getValue()
```

Return the memory cached object, or null if not found or has expired.

Returns:

- the memory cached object, or null if not found or has expired.

---

setValue

```
public void setValue(Object value)
```

Set the value to cache in memory with the specified key and cacheTimeout properties.

Parameters:

- value - the value to set in the cache for the specified key and cacheTimeout property

Path

Package:

- com.avoka.tm.util

Class Path

- java.lang.Object
- com.avoka.tm.util.Path

```
public class Path
extends Object
```

Provides an object path navigation class to return the value for the given expression. This class will automatically select the appropriate path navigation library for the given source object which may be JSON text, XML text, XML document or Java POJO (Plain Old Java Object).

Path Sources

The supported `Path` sources can be constructed by JSON text, XML text, XML document or Java POJO.

Expressions

Here is a complete overview and a side by side comparison of the JSONPath syntax elements with its XPath counterparts.

XPath	JSONPath	Description
/	$	the root object/element
.	@	the current object/element
/	. or []	child operator
..	n/a	parent operator
//	..	recursive descent. JSONPath borrows this syntax from E4X.
*	*	wildcard. All objects/elements regardless their names.
@	n/a	attribute access. JSON structures don't have attributes.
[]	[]	subscript operator. XPath uses it to iterate over element collections and for predicates. In Javascript and JSON it is the native array operator.
|	[,]	Union operator in XPath results in a combination of node sets. JSONPath allows alternate names or array indices as a set.
n/a	[start:end:step]	array slice operator borrowed from ES4.
[]	?()	applies a filter (script) expression.
n/a	()	script expression, using the underlying script engine.
()	n/a	grouping in Xpath

JSON Sources

With JSON sources the expression language used is JSONPath. When using the `val(exp)` method this class will return the text content value for the first selected node, or null if no node was found. The expression operators include $(root node of the structure), @ (current node that is being processed, e.g. "user[?(@.firstName==John)]") and * (wildcard indicating all elements in scope, e.g. user[*]). See JayWay JSONPath Library Guide for more documentation.

Object Sources

With Object sources the expression language used is "property names", e.g. "user.firstName". When using the `val(exp)` method this class will return the text content value for the first selected node, or null if no node was found.

XML Sources

Note that the implementation is namespace aware but ignores default namespace's override.

Examples

JSON Examples

This Groovy example creates Path from JSON string, and then reads out a value from path.

```
import com.avoka.tm.util.Path

String json =
'''{
   "address": {
     "line1": "1 Street",
     "postCode": 2000
   }
}'''

Path path = new Path(json)

def line1 = path.val('address.line1')
def postCode = path.val('$.[\'address\'].[\'postCode\']')
```

Note you have the full power of JSON Path so you can perform like:

```
def addressNodeFiltered = path.val("address[?(@.postCode == 2000)]")
def allAddressChildrenValuesList = path.val("address[*]")
```

Java Object Examples

This Groovy example creates Path from Object, and then reads out a value from path.

```
import com.avoka.tm.util.Path

User user = new User()
user.setFirstName("George")

ContactDetails cd = new ContactDetails()
cd.setEmail("george@email.com")
user.setContactDetails(cd)

Address address = new Address()
address.setCity("Sydney")
address.setPostCode(2153)
user.getAddressList().add(address)

Path path = new Path(user)

def userFirstName = path.val("firstName")
def userEmail = path.val("contactDetails.email")
def postCode = path.val("addressList[0].postCode")
```

XML Examples

This Groovy example creates Path from XML string, and then reads out a value from path.

```
import com.avoka.tm.util.Path

//  source - XML text
String formXml =
'''<AvokaSmartForm>
   <YourDetails>
      <FirstName> ... </FirstName>
      <LastName> ... </LastName>
      <Email> ... </Email>
   </YourDetails>
</AvokaSmartForm>'''

Path path = new Path(formXml)
def userFirstName = path.val("//FirstName")
def userEmail = path.val("/AvokaSmartForm/YourDetails/Email")

// schemaSeed - XML Document source
Path path2 = new Path(schemaSeed)
def userEmail = path2.val("//Email")
```

Note you have the full power of XQuery so you can perform XQuerys like:

```
def countryList = path.val("//root/country[@value='USA']")
```

The Path class also provides a `nodes()` method which will return a typed List of XML Nodes which can be iterated over. In contrast the Path `val()` method will return the text content of the first selected XML Node.

```
import com.avoka.tm.util.Path
import org.w3c.dom.Node

def xml =
'''<bar>
    <drink type="Beer"> 4 Pines </drink>
    <drink type="Beer"> Coopers </drink>
    <drink type="Wine"> Cab Sav </drink>
</bar>'''

// Select first value
Path path = new Path(xml)

def firstDrink = path.val('//drink')

println 'firstDrink value: ' + firstDrink + ', type: ' + firstDrink.class.simpleName

// Select node list
List<Node> drinks = path.nodes('//drink')

for (drink in drinks) {
    println 'textContent: ' + drink.textContent + ', type: ' + drink.class.simpleName + ', toString: ' + drink
}
```

Script output results:

```
firstDrink value: 4 Pines, type: String

textContent: 4 Pines, type: DeferredElementImpl, toString: <?xml version="1.0" encoding="UTF-8"?> <drink type="
Beer">4 Pines</drink>
textContent: Coopers, type: DeferredElementImpl, toString: <?xml version="1.0" encoding="UTF-8"?> <drink type="
Beer">Coopers</drink>
textContent: Cab Sav, type: DeferredElementImpl, toString: <?xml version="1.0" encoding="UTF-8"?> <drink type="
Wine">Cab Sav</drink>
```

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`Path(Object source)` Create a path evaluation object with the given source.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`List<Node>`	`nodes(String path)` Return the list of XML Node values for the given expression.
`List<Path>`	`paths(String path)` Return the list of paths for the given expression.
`Object`	`val(String path)` Return the value for the given expression.

`List<Object>`	`vals(String path)` Return the list of values for the given expression.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

Path

```
public Path(Object source)
```

Create a path evaluation object with the given source.

Parameters:

- `source` - the source

Method Detail

val

```
public Object val(String path)
```

Return the value for the given expression.

With XML sources the expression language used is XPath. When using the `val(exp)` method this class will return the text content value for the first selected node, or null if no node was found.

Parameters:

- `path` - the path expression to evaluate (required)

Returns:

- value for the path expression, otherwise `null` if path is invalid or can not be found.

---

vals

```
public List<Object> vals(String path)
```

Return the list of values for the given expression.

With an XML source the expression language used is XPath. When using the `vals(exp)` method this class will return the text content value for the list of selected nodes, or an empty list if not found.

Parameters:

- `path` - the path expression to evaluate (required)

Returns:

- the list of values for the given expression, otherwise empty `List` if can not be found.

---

nodes

```
public List<Node> nodes(String path)
```

Return the list of XML Node values for the given expression.

This method will throw an UnsupportedOperationException if the Path source is not an XML source.

Parameters:

- `path` - the path expression to evaluate (required)

Returns:

- the list of DOM Node values for the given the path expression

---

paths

```
public List<Path> paths(String path)
```

Return the list of paths for the given expression.

Parameters:

- `path` - the path expression to evaluate (required)

Returns:

- the list of values for the given expression.

RedirectException

Package:

- com.avoka.tm.util

Class RedirectException

- java.lang.Object
- java.lang.Throwable
- java.lang.Exception
- java.lang.RuntimeException
- com.avoka.fc.core.servlet.RedirectException
- com.avoka.tm.util.RedirectException

All Implemented Interfaces:

- Serializable

```
public class RedirectException
extends com.avoka.fc.core.servlet.RedirectException
```

Provides an redirect exception which can be used to redirect the user to another location.

By convention if the redirect target URL is prefixed with '/' then the web applications context path will be prepended to the target URL. For example:

```
'/form-expired.htm' -> 'http://www.mycorp.com/forms/form-expired.htm'
```

Note this URL context path pre-pending should be done by the calling code and will not be done by this class.

Since:

- 5.0

See Also:

- Serialized Form

Constructor Summary

Constructors

Constructor and Description
`RedirectException(String target)` Create a Redirect exception with the given target URL.

Method Summary

Methods inherited from class com.avoka.fc.core.servlet.RedirectException

`getTarget, toString`

Methods inherited from class java.lang.Throwable

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace`

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

RedirectException

```
public RedirectException(String target)
```

Create a Redirect exception with the given target URL.
By convention if the redirect target URL is prefixed with '/' then the web applications context path will be prepended to the target URL. For example:

```
'/form-expired.htm' -> 'http://www.mycorp.com/forms/form-expired.htm'
```

Note this URL context path pre-pending should be done by the calling code and will not be done by this class.

Parameters:

- `target` - the redirect target URL

Security

Package:

- com.avoka.tm.util

Class Security

- java.lang.Object
- com.avoka.tm.util.Security

```
public class Security
extends Object
```

Provides transaction security functions.

Examples

Please find the transaction security function examples below.

XML Safe Check Example

This Groovy example shows how to check if the XML document does not contain any XSS characters.

```
import com.avoka.tm.util.Security

boolean isSafe = Security.isXmlDocumentSafe(xmlDocument)
```

Text Safe Check Example

This Groovy example shows how to check if the XML document does not contain any XSS characters.

```
import com.avoka.tm.util.Security

boolean isSafe = Security.isXmlTextSafe(xmlText)
```

Adding Transaction Security Tokens to Session Example

This Groovy example shows how to add the transaction security tokens to the session.

```
import com.avoka.tm.util.Security

Security.addSessionTxnSecurityTokens(request, txnId)
```

Since:

- 5.0

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method and Description
static void	addSessionTxnSecurityTokens(HttpServletRequest request, Txn txn) Add the transaction security tokens to the session.
static boolean	isXssSafeFilename(String value) Return true if the filename value is XSS safe.
static boolean	isXssSafeText(String text) Return true if the XML text does not contain any XSS characters, or false otherwise.
static boolean	isXssSafeXmlDoc(Document document)

	Return true if the XML document does not contain any XSS characters, or false otherwise.
`static boolean`	`isXssSafeXmlText(String value)` Return true if the XML text does not contain any XSS characters, or false otherwise.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Method Detail

addSessionTxnSecurityTokens

```
public static void addSessionTxnSecurityTokens(HttpServletRequest request,
                                               Txn txn)
```

Add the transaction security tokens to the session.

Parameters:

- `request` - the HTTP request (required)
- `txn` - the transaction (required)

---

isXssSafeXmlDoc

```
public static boolean isXssSafeXmlDoc(Document document)
```

Return true if the XML document does not contain any XSS characters, or false otherwise. Please see dangerous characters:

XSS Filter Evasion Cheat Sheet

Parameters:

- `document` - the XML document to test (required)

Returns:

- true, if is XSS safe or false otherwise

---

isXssSafeXmlText

```
public static boolean isXssSafeXmlText(String value)
```

Return true if the XML text does not contain any XSS characters, or false otherwise. Please see dangerous characters:

XSS Filter Evasion Cheat Sheet

Parameters:

- `value` - the XML text to test (required)

Returns:

- true, if is XSS safe or false otherwise

---

isXssSafeText

```
public static boolean isXssSafeText(String text)
```

Return true if the XML text does not contain any XSS characters, or false otherwise. Please see dangerous characters:

XSS Filter Evasion Cheat Sheet

Parameters:

- `text` - the XML text to test (required)

Returns:

- true, if is XSS safe or false otherwise

---

isXssSafeFilename

```
public static boolean isXssSafeFilename(String value)
```

Return true if the filename value is XSS safe. This applies the standard XSS checks except 'document.' and 'window.' values. Please see dangerous characters:

XSS Filter Evasion Cheat Sheet

Parameters:

- `value` - the value to test (required)

Returns:

- true, if is XSS safe

Threads

Package:

- com.avoka.tm.util

Class Threads

- java.lang.Object
- com.avoka.tm.util.Threads

```
public final class Threads
extends Object
```

Provides Thread functions.

Examples

Please find the thread functions examples below.

Thread Sleep Example

This Groovy example shows how to sleep the thread for 200ms.

```
import com.avoka.tm.util.Threads

// perform web service request
...

// sleep 200ms
Threads.sleep(200)

// continue
...
```

Since:

- 5.0.1

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method and Description
`static void`	`sleep(long millis)` Causes the currently executing thread to sleep for the specified number of milliseconds (max 30,000ms).

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Method Detail

---

sleep

```
public static void sleep(long millis)
```

Causes the currently executing thread to sleep for the specified number of milliseconds (max 30,000ms).

Parameters:

- `millis` - the length of time to sleep in milliseconds (between 0 and 30,000).

See Also:

- `Thread.sleep(long)`

TxnUrlBuilder

Package:

- com.avoka.tm.util

Class TxnUrlBuilder

- java.lang.Object
- com.avoka.tm.util.TxnUrlBuilder

```
public class TxnUrlBuilder
extends Object
```

Provides a transaction form and PDF receipt URL builder class.

Examples

Open Form URL

A open form URL generation example.

```
import com.avoka.tm.util.*

String formUrl = new TxnUrlBuilder()
   .setTxn(txn)
   .setSpaceName("Work Space")
   .buildFormUrl()
```

PDF Receipt URL

A PDF receipt URL generation example.

```
import com.avoka.tm.util.*

String receiptUrl = new TxnUrlBuilder()
   .setTxn(txn)
   .setSpaceName("Work Space")
   .buildReceiptUrl()
```

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`TxnUrlBuilder()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`String`	`buildFormUrl()` Return the open form URL.
`String`	`buildReceiptUrl()` Return the open PDF receipt URL.
`TxnUrlBuilder`	`setSpaceName(String spaceName)` Set the space name property.

TxnUrlBuilder	setTxn(Txn txn) Set the transaction property.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

TxnUrlBuilder

```
public TxnUrlBuilder()
```

Method Detail

setTxn

```
public TxnUrlBuilder setTxn(Txn txn)
```

Set the transaction property.

Parameters:

- `txn` - the transaction property

Returns:

- the transaction URL builder

---

setSpaceName

```
public TxnUrlBuilder setSpaceName(String spaceName)
```

Set the space name property.

Parameters:

- `spaceName` - the space name property

Returns:

- the transaction URL builder

---

buildFormUrl

```
public String buildFormUrl()
```

Return the open form URL.

Returns:

- the open form URL

---

buildReceiptUrl

```
public String buildReceiptUrl()
```

Return the open PDF receipt URL.

Returns:

- the open PDF receipt URL

VelTemplate

Package:

- com.avoka.tm.util

Class VelTemplate

- java.lang.Object
- com.avoka.tm.util.VelTemplate

```
public class VelTemplate
extends Object
```

Provides a Apache Velocity templating class.

Examples

Email Template from Service Parameter

Provides an example creating an email body message using a Velocity template provided as a service parameter.

```
import com.avoka.tm.util.*

String emailTemplate = svcDef.paramsMap["emailTemplate"]
User user = ...

String emailMessage  = new VelTemplate()
     .setTemplate(emailTemplate)
     .addModelValue("user", user)
     .merge()
```

Email Template from Organization Property

Provides an example creating an email body message using a Velocity template provided from an Organization property.

```
import com.avoka.tm.query.*
import com.avoka.tm.util.*

String emailTemplate = new PropertyQuery()
     .setName("HTML Email Welcome")
     .setClientCode("maguire")
     .getValue()

User user = ...

String emailMessage  = new VelTemplate()
     .setTemplate(emailTemplate)
     .addModelValue("user", user)
     .merge()
```

Since:

- 5.0.0

See Also:

- PropertyQuery

Constructor Summary

Constructors

Constructor and Description
`VelTemplate()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
VelTemplate	addModelValue(String key, Object value) Set the model key and value to merge with the template.
String	merge() Merge the velocity template with the model and return the evaluated text value.
VelTemplate	setModel(Map<String,Object> model) Set the model value to merge with the template.
VelTemplate	setTemplate(String template) Set the Velocity template value.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

VelTemplate

```
public VelTemplate()
```

Method Detail

setTemplate

```
public VelTemplate setTemplate(String template)
```

Set the Velocity template value.

Parameters:

- `template` - the velocity template value

Returns:

- the vel template

---

setModel

```
public VelTemplate setModel(Map<String,Object> model)
```

Set the model value to merge with the template.

Parameters:

- `model` - the model value to merge with the template (required)

Returns:

- the vel template

---

addModelValue

```
public VelTemplate addModelValue(String key,
                                 Object value)
```

Set the model key and value to merge with the template.

Parameters:

- `key` - the model key to add (required)
- `value` - the model value to add (required)

Returns:

- the vel template

---

merge

```
public String merge()
```

Merge the velocity template with the model and return the evaluated text value.

Returns:

- the evaluated text value from merging the Velocity template with the model

XmlDoc

Package:

- com.avoka.tm.util

Class XmlDoc

- java.lang.Object
- com.avoka.tm.util.XmlDoc

```
public class XmlDoc
extends Object
```

Provides XML Document class for use in Groovy scripts. An XmlDoc object wraps an underlying XML Document object and provides convenience methods for performing access and modification operations.

Examples

Please find the XML Document examples below.

XML Read Example

This Groovy example creates an XmlDoc from a delivery formXml string value, and then reads out a values and puts them in a map.

```
import com.avoka.tm.util.*

XmlDoc xmlDoc = new XmlDoc(formXml)

Map contact = [:]
contact.firstName = xmlDoc.getText('//YourDetails/FirstName')
contact.lastName = xmlDoc.getText('//YourDetails/LastName')
contact.email = xmlDoc.getText('//YourDetails/Email')
contact.state = xmlDoc.getText('//Location/State')
```

Please see the W3C Schools XPath Syntax for more examples.

XML Write Example

This Groovy example writes form prefill data into a formXml XML Document.

```
import com.avoka.tm.util.*

XmlDoc xmlDoc = new XmlDoc(formXml)

def userDetails = ...

xmlDoc.setText('/AvokaSmartForm/YourDetails/FirstName', userDetails.firstName)
xmlDoc.setText('/AvokaSmartForm/YourDetails/LastName', userDetails.lastName)
xmlDoc.setText('/AvokaSmartForm/YourDetails/Email', userDetails.email)
```

You can also add Nodes or a list of Nodes to an XML Document using the `addNode(String, Node)` or `addNodeList(String, List)` methods. See the Composer Cascading DD List Example later to see this in action.

Adding XML Content as Text Example

When creating large amounts of XML it is often easer to use string templating techniques to create the XML content and then add this to the target XML document.

The example below is using the `addContent(String, String, String)` method to add the `'/ItemList/Item'` elements from the source `prefillDataContext` XML text to the `xmlDoc` object under the node `'/AvokaSmartForm/Accounts'`.

```
import com.avoka.tm.util.*

// Text prefill XML content
String prefillDataContent = '<ItemList> <Item>A1</Item> <Item>B2</Item> </ItemList>'
```

```
// Target document to add prefill content to
XmlDoc xmlDoc = new XmlDoc(formXml)
xmlDoc.addContent(prefillDataContent, '/ItemList/Item', '/AvokaSmartForm/Accounts')
```

If you need to replace existing content in an XML Document rather than adding to it, then use the `removeNodes(String)` method first before adding the content. For example:

```
import com.avoka.tm.util.*

String prefillDataContent = '<ItemList> <Item>A1</Item> <Item>B2</Item> </ItemList>'

XmlDoc xmlDoc = new XmlDoc(formXml)

// Remove any existing data
xmlDoc.removeNodes('/AvokaSmartForm/Accounts')

xmlDoc.addContent(prefillDataContent, '/ItemList/Item', '/AvokaSmartForm/Accounts')
```

See http://en.wikibooks.org/wiki/XQuery for information on how to use XQuery.

Since:

- 5.0.0

Constructor Summary

Constructors

Constructor and Description
`XmlDoc(Object source)` Create an XmlDoc object from the given XML source, either String, Document, InputStream or byte[].

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`Node`	`addContent(String sourceContent, String sourceXPath, String targetXPath)` Add the given XML content from the source XPath to this XML Document object at the specified target XPath.
`Node`	`addNode(String xpath, Node node)` Add the given Node to the XML Document object at to specified XPath.
`void`	`addNodeList(String xpath, List<Node> nodeList)` Add the given Node list to the XML Document object at to specified XPath.
`String`	`evalXPath(String xpath)` Evaluate the XPath expression and return the given string value.
`Map<String,String>`	`getDataMap(String jsonMap)` Return a map of XML data element values from the document using the given jsonMap definition.
`Document`	`getDocument()` Return the underlying XML Document object.
`Node`	`getNode(String xpath)` Return the XML node for the given XPath.
`List<Node>`	`getNodeList(String xpath)` Return the list of XML Nodes for the given XPath expression.
`String`	`getText(String xpath)` Return the text value for the given XPath or null if not found.
`boolean`	`removeNodeChildren(String xpath)` Remove the child nodes from the XML Document element specified by the XPath.
`boolean`	`removeNodes(String xpath)` Remove the nodes from the XML Document object specified by the XPath.
`Node`	`setNode(String xpath)`

	Create the XML elements specified by the XPath expression if they don't exist, and return the leaf node.
Node	setText(String xpath, String textContent) Set the content text on the XML element specified by the XPath expression.
String	toFormattedString() Return the formatted string representation of the Document.
String	toString() Return the string representation of the Document.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

XmlDoc

```
public XmlDoc(Object source)
```

Create an XmlDoc object from the given XML source, either String, Document, InputStream or byte[].

Parameters:

- source - the source XML, either String, Document, InputStream or byte[].

Method Detail

addContent

```
public Node addContent(String sourceContent,
                       String sourceXPath,
                       String targetXPath)
```

Add the given XML content from the source XPath to this XML Document object at the specified target XPath.

Parameters:

- sourceContent - the source XML content to add to the XML Document
- sourceXPath - the XPath to the node within the source XML content to add
- targetXPath - the XPath to add the content to in this XML Document

Returns:

- the Node of the target XPath

---

addNode

```
public Node addNode(String xpath,
                    Node node)
```

Add the given Node to the XML Document object at to specified XPath. If the XPath nodes do not exist in the document they will be created. This method will return a deep cloned copy of the given Node which was added to the XML Document at the specified path. XPath expressions must use an absolute path to avoid ambiguity when creating XML path elements, using // search criteria is not permitted.

Parameters:

- xpath - the XPath location of where to add the node
- node - the Node to add to the XML Document

Returns:

- the cloned node added to the XML Document at the specified XPath.

---

addNodeList

```
public void addNodeList(String xpath,
                        List<Node> nodeList)
```

Add the given Node list to the XML Document object at to specified XPath. If the XPath nodes do not exist in the document they will be created. This method will return a deep cloned copy of the given list of Node which was added to the XML Document at the specified path. XPath expressions must use an absolute path to avoid ambiguity when creating XML path elements, using // search criteria is not permitted.

Parameters:

- xpath - the XPath location of where to add the nodes
- nodeList - the list of Nodes to add to the XML Document

---

getDocument

`public Document getDocument()`

Return the underlying XML Document object.

Returns:

- the underlying XML Document object.

---

getNode

`public Node getNode(String xpath)`

Return the XML node for the given XPath.

Parameters:

- `xpath` - the XPath expression of the node

Returns:

- the node for the given xpath

---

getNodeList

`public List<Node> getNodeList(String xpath)`

Return the list of XML Nodes for the given XPath expression.

Parameters:

- `xpath` - the XPath expression

Returns:

- the list of XML Nodes for the given XPath expression

---

getText

`public String getText(String xpath)`

Return the text value for the given XPath or null if not found.

```
def xmlDoc = new com.avoka.component.xml.XmlDoc(submissionXml)

def contact = [:]
contact.firstName = xmlDoc.getText('//YourDetails/FirstName')
contact.lastName = xmlDoc.getText('//YourDetails/LastName')
contact.email = xmlDoc.getText('//YourDetails/Email')
contact.state = xmlDoc.getText('//Location/State')
```

Parameters:

- `xpath` - the XPath to search

Returns:

- the text value for the given XPath or null if not found

---

setNode

`public Node setNode(String xpath)`

Create the XML elements specified by the XPath expression if they don't exist, and return the leaf node.

XPath expressions must use an absolute path to avoid ambiguity when creating XML path elements, using // search criteria is not permitted.

```
def line1 = xmlDoc.set('/AvokaSmartForm/YourDetails/Address/Line1")
```

Parameters:

- `xpath` - the XML elements to create

Returns:

- the node specified by the XPath if found or the created XML node

---

setText

```
public Node setText(String xpath,
                    String textContent)
```

Set the content text on the XML element specified by the XPath expression. This method will create the elements specified by the path if they don't exist.

XPath expressions must use an absolute path to avoid ambiguity when creating XML path elements, using // search criteria is not permitted.

```
xmlDoc.setText('/AvokaSmartForm/YourDetails/Address/Line1", "30 George Street")
```

Example

```
def xmlDoc = new com.avoka.component.xml.XmlDoc(schemaSeed)

def userDetails = ...

xmlDoc.setText('/AvokaSmartForm/YourDetails/FirstName', userDetails.firstName)
xmlDoc.setText('/AvokaSmartForm/YourDetails/LastName', userDetails.lastName)
xmlDoc.setText('/AvokaSmartForm/YourDetails/Email', userDetails.email)
```

Parameters:

- `xpath` - the XML element to set the text on, this path will be created if it does not exist
- `textContent` - the element content text value to set

Returns:

- the node specified by the XPath if found or the created XML node

---

evalXPath

```
public String evalXPath(String xpath)
```

Evaluate the XPath expression and return the given string value.

Example

```
def xmlDoc = new XmlDoc(schemaSeed)

def fullname = xmlDoc.evalXPath("concat(//FirstName, ' ',//LastName)")
```

Parameters:

- `xpath` - the XPath expression (required)

Returns:

- the evaluated XPath expression as a string value

Since:

- 4.0.0

---

removeNodes

```
public boolean removeNodes(String xpath)
```

Remove the nodes from the XML Document object specified by the XPath.

Parameters:

- `xpath` - the XPath of the nodes to remove

Returns:

- true if a one or more nodes were removed from the XML Document object

---

removeNodeChildren

```
public boolean removeNodeChildren(String xpath)
```

Remove the child nodes from the XML Document element specified by the XPath.

Parameters:

- `xpath` - the XPath of the parent node whose children should be removed (required)

Returns:

- true if a one or more nodes were removed from the XML Document object

---

getDataMap

```
public Map<String,String> getDataMap(String jsonMap)
```

Return a map of XML data element values from the document using the given jsonMap definition.
Example JSON mapping format:

```
{
   "firstName": "//SmartForm/FormData/Contact/FirstName",
   "lastName": "//SmartForm/FormData/Contact/LastName",
   "email": "//SmartForm/FormData/Contact/Email"
}
```

This method provides a convenience wrapper around `XmlUtils.getXmlDataMap(Document, String)`.

Parameters:

- `jsonMap` - the JSON format name to XPath mapping definition (required)

Returns:

- a map of XML data element values from the given document using the jsonMap definition

---

toFormattedString

```
public String toFormattedString()
```

Return the formatted string representation of the Document.

Returns:

- the formatted string representation of the Document

---

toString

```
public String toString()
```

Return the string representation of the Document.

Overrides:

- `toString` in class `Object`

Returns:

- the string representation of the Document

com.avoka.tm.vo

Classes in Package com.avoka.tm.vo

- FileAttach
- Form
- Job
- JobAction
- JobStep
- Space
- SvcConn
- SvcDef
- Txn
- TxnCheckpoint
- User

FileAttach

Package:

- com.avoka.tm.vo

Class FileAttach

- java.lang.Object
- com.avoka.tm.vo.FileAttach

```
public class FileAttach
extends Object
```

Provide a Transaction File Attachment value object class.

Since:

- 5.0.0

Field Summary

Fields

Modifier and Type	Field and Description
String	attachKey The required attachment key.
String	attachName The required attachment name.
String	contentType The file attachment content type.
Boolean	dataDeleted The file attachment data has been deleted.
byte[]	fileData The attachment file data.
String	fileDataHash The attachment file data hash (SHA256).
String	fileName The attachment file name.
Long	fileSize The attachment file data size in bytes.
Long	id The attachment id (PK).
String	submitMethod The attachment submission method [Electronic | Manual].
Date	timeUploaded The time the attachment was uploaded.
String	userDesc The user description.
String	userLoginName The user login name (username).
String	virusStatus The virus scan status.

Constructor Summary

Constructors

Constructor and Description

`FileAttach(com.avoka.fc.core.entity.Attachment attachment)` Create a file attachment value object.
`FileAttach(Map fields)` Create a unit testing FileAttach value object with the given fields.
`FileAttach(String attachmentName, String fileName, byte[] fileData)` Create a file attachment value object with the given attachment name, file name and file data.
`FileAttach(String attachmentName, String fileName, byte[] fileData, String userDescription)` Create a file attachment value object with the given attachment name, file name, file data and user description.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`String`	`toString()`

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Field Detail

id

`public final Long id`

The attachment id (PK).

attachKey

`public final String attachKey`

The required attachment key.

attachName

`public final String attachName`

The required attachment name.

submitMethod

`public final String submitMethod`

The attachment submission method [Electronic | Manual].

dataDeleted

`public final Boolean dataDeleted`

The file attachment data has been deleted.

fileName

`public final String fileName`

The attachment file name.

fileData

`public final byte[] fileData`

The attachment file data.

fileDataHash

`public final String fileDataHash`

The attachment file data hash (SHA256).

fileSize

```
public final Long fileSize
```

The attachment file data size in bytes.

---

timeUploaded

```
public final Date timeUploaded
```

The time the attachment was uploaded.

---

contentType

```
public final String contentType
```

The file attachment content type.

---

userDesc

```
public final String userDesc
```

The user description.

---

userLoginName

```
public final String userLoginName
```

The user login name (username).

---

virusStatus

```
public final String virusStatus
```

The virus scan status.

Constructor Detail

FileAttach

```
public FileAttach(com.avoka.fc.core.entity.Attachment attachment)
```

Create a file attachment value object.

Parameters:

- `attachment` - the transaction attachment entity (required)

FileAttach

```
public FileAttach(String attachmentName,
                  String fileName,
                  byte[] fileData,
                  String userDescription)
```

Create a file attachment value object with the given attachment name, file name, file data and user description.

Parameters:

- `attachmentName` - the required attachment name (required)
- `fileName` - the file name of the attachment (required)
- `fileData` - the attachment file data (required)
- `userDescription` - the user description of the file attachment (optional)

FileAttach

```
public FileAttach(String attachmentName,
                  String fileName,
                  byte[] fileData)
```

Create a file attachment value object with the given attachment name, file name and file data.

Parameters:

- `attachmentName` - the required attachment name (required)
- `fileName` - the file name of the attachment (required)
- `fileData` - the attachment file data (required)

FileAttach

`public FileAttach(Map fields)`

Create a unit testing FileAttach value object with the given fields.

Parameters:

- `fields` - the submission entity fields (required)

Since:

- 5.1.4

Method Detail

toString

`public String toString()`

Overrides:

- `toString` in class `Object`

Returns:

- a string representation of the object.

Form

Package:

- com.avoka.tm.vo

Class Form

- java.lang.Object
- com.avoka.tm.vo.Form

```
public class Form
extends Object
```

Provide a Form value object class.

Since:

- 5.0.0

Field Summary

Fields

Modifier and Type	Field and Description
`String`	`description` The current form versions description property.
`String`	`formCode` The form code.
`String`	`formName` The form name.
`Long`	`formVersionId` The current form version id (PK).
`Set<String>`	`groupNames` The set of associated form group group names.
`Long`	`id` The form id (PK).
`Long`	`orgId` The organization client id (PK).
`Set<String>`	`spaceNames` The set of associated space names.
`Set<String>`	`versions` The set of associated versions.

Constructor Summary

Constructors

Constructor and Description
`Form(com.avoka.fc.core.entity.Form form)` Create a Form value object with the given form entity parameter.
`Form(Map fields)` Create a unit testing Form value object with the given fields.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`String`	`toString()`

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Field Detail

id

`public final Long id`

The form id (PK).

formCode

`public final String formCode`

The form code.

formName

`public final String formName`

The form name.

orgId

`public final Long orgId`

The organization client id (PK).

formVersionId

`public final Long formVersionId`

The current form version id (PK).

description

`public final String description`

The current form versions description property.

groupNames

`public final Set<String> groupNames`

The set of associated form group group names.

spaceNames

`public final Set<String> spaceNames`

The set of associated space names.

versions

`public final Set<String> versions`

The set of associated versions.

Since:

- 5.1.0

Constructor Detail

Form

`public Form(com.avoka.fc.core.entity.Form form)`

Create a Form value object with the given form entity parameter.

Parameters:

- `form` - the form entity parameter (required)

Form

`public Form(Map fields)`

Create a unit testing Form value object with the given fields.

Parameters:

- `fields` - the submission entity fields (required)

Since:

- 5.1.4

Method Detail

toString

`public String toString()`

Overrides:

- `toString` in class `Object`

Returns:

- a string representation of the object.

Job

Package:

- com.avoka.tm.vo

Class Job

- java.lang.Object
- com.avoka.tm.vo.Job

```
public class Job
extends Object
```

Provide a Job value object class.

Since:

- 5.0.0

Field Summary

Fields

Modifier and Type	Field and Description
`String`	`clientCode` The organization client code.
`Long`	`controllerServiceId` The job controller service id.
`Long`	`currentJobStepId` The current job step id (PK).
`Long`	`id` The job record id (PK).
`String`	`jobKey` The job key.
`List<JobStep>`	`jobSteps` The list of job steps.
`String`	`name` The job name.
`Long`	`orgId` The organization id (PK).
`Map<String,String>`	`propertyMap` The job property map.
`String`	`referenceNumber` The job reference number.
`String`	`status` The job status.
`static String`	`STATUS_CANCELLED` The job "Cancelled" status.
`static String`	`STATUS_COMPLETED` The job "Completed" status.
`static String`	`STATUS_ERROR` The job "Error" status.
`static String`	`STATUS_EXPIRED` The job "Expired" status.
`static String`	`STATUS_IN_PROGRESS` The job "In Progress" status.

static String	STATUS_PAUSED The job "Paused" status.
Date	timeCompletionScheduled The time the job is scheduled for completion.
Date	timeCreated The time the job was created.
Date	timeFinished The time the job was finished.
Date	timeLastProcessed The time the job was last processed.
Date	timePurgeScheduled The time the job is scheduled to be purged.
List<Long>	txnIdList The list of job transaction ids.

Constructor Summary

Constructors

Constructor and Description
Job(com.avoka.fc.core.entity.Job job) Create a Job value object with the given job entity parameter.
Job(Map fields) Create a unit testing Job value object with the given fields.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
String	toString()

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

STATUS_CANCELLED

```
public static final String STATUS_CANCELLED
```

The job "Cancelled" status.

See Also:

- Constant Field Values

STATUS_COMPLETED

```
public static final String STATUS_COMPLETED
```

The job "Completed" status.

See Also:

- Constant Field Values

STATUS_ERROR

```
public static final String STATUS_ERROR
```

The job "Error" status.

See Also:

- Constant Field Values

STATUS_EXPIRED

`public static final String STATUS_EXPIRED`

The job "Expired" status.

See Also:

- Constant Field Values

STATUS_IN_PROGRESS

`public static final String STATUS_IN_PROGRESS`

The job "In Progress" status.

See Also:

- Constant Field Values

STATUS_PAUSED

`public static final String STATUS_PAUSED`

The job "Paused" status.

See Also:

- Constant Field Values

---

id

`public final Long id`

The job record id (PK).

---

name

`public final String name`

The job name.

---

orgId

`public final Long orgId`

The organization id (PK).

---

clientCode

`public final String clientCode`

The organization client code.

---

controllerServiceId

`public final Long controllerServiceId`

The job controller service id.

---

jobKey

`public final String jobKey`

The job key.

---

referenceNumber

`public final String referenceNumber`

The job reference number.

---

status

`public final String status`

The job status.

---

currentJobStepId

`public final Long currentJobStepId`

The current job step id (PK).

---

timeCompletionScheduled

`public final Date timeCompletionScheduled`

The time the job is scheduled for completion.

---

timeCreated

`public final Date timeCreated`

The time the job was created.

---

timeFinished

`public final Date timeFinished`

The time the job was finished.

---

timeLastProcessed

`public final Date timeLastProcessed`

The time the job was last processed.

---

timePurgeScheduled

`public final Date timePurgeScheduled`

The time the job is scheduled to be purged.

---

txnIdList

`public List<Long> txnIdList`

The list of job transaction ids.

---

jobSteps

`public List<JobStep> jobSteps`

The list of job steps.

---

propertyMap

`public final Map<String,String> propertyMap`

The job property map.

Constructor Detail

Job

`public Job(com.avoka.fc.core.entity.Job job)`

Create a Job value object with the given job entity parameter.

Parameters:

- `job` - the job entity parameter (required)

Job

`public Job(Map fields)`

Create a unit testing Job value object with the given fields.

Parameters:

- `fields` - the submission entity fields (required)

Since:

- 5.1.4

Method Detail

toString

`public String toString()`

Overrides:

- `toString` in class `Object`

Returns:

- a string representation of the object.

JobAction

Package:

- com.avoka.tm.vo

Class JobAction

- java.lang.Object
- com.avoka.tm.vo.JobAction

```
public class JobAction
extends Object
```

Provide a Job Action value object class.

Since:

- 5.0.0

Field Summary

Fields

Modifier and Type	Field and Description
`Integer`	`actionAttempts` The job action attempts.
`Integer`	`actionMaxAttempts` The job action maximum attempts.
`String`	`actionMessage` The job action message.
`Long`	`actionServiceId` The job action service definition id (PK).
`Integer`	`assignRepeatIndex` The assign repeat index.
`String`	`assignRepeatItem` The assign repeat item.
`String`	`externalReferenceNumber` The job action external reference number.
`Long`	`id` The job action record (PK).
`String`	`jobAtionKey` The job action key.
`String`	`name` The job action name.
`String`	`routeResult` The job action route result.
`Integer`	`sequence` The job action sequence.
`String`	`status` The job action status.
`static String`	`STATUS_ASSIGNED` The job step "Assigned" status.
`static String`	`STATUS_CANCELLED` The job step "Cancelled" status.
`static String`	`STATUS_COMPLETED` The job step "Completed" status.

static String	STATUS_ERROR The job step "Error" status.
static String	STATUS_EXPIRED The job step "Expired" status.
static String	STATUS_IN_PROGRESS The job step "In Progress" status.
static String	STATUS_INVOKED The job step "Invoked" status.
static String	STATUS_PENDING The job step "Pending" status.
static String	STATUS_READY The job step "Ready" status.
Date	timeActionExecuted The time action was last executed.
Date	timeActionNextAttempt The time action should next be attempted.
Date	timeCreated The time the job action was created.
Date	timeFinished The time the job action was finished.
Long	txnId The action transaction id (PK).
String	type The job action type.

Constructor Summary

Constructors

Constructor and Description
JobAction(com.avoka.fc.core.entity.JobAction jobAction) Create a Job Action value object with the given job action entity parameter.
JobAction(Map fields) Create a unit testing JobAction value object with the given fields.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
String	toString()

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

STATUS_ASSIGNED

```
public static final String STATUS_ASSIGNED
```

The job step "Assigned" status.

See Also:

- Constant Field Values

STATUS_CANCELLED

```
public static final String STATUS_CANCELLED
```

The job step "Cancelled" status.

See Also:

- Constant Field Values

STATUS_COMPLETED

`public static final String STATUS_COMPLETED`

The job step "Completed" status.

See Also:

- Constant Field Values

STATUS_ERROR

`public static final String STATUS_ERROR`

The job step "Error" status.

See Also:

- Constant Field Values

STATUS_EXPIRED

`public static final String STATUS_EXPIRED`

The job step "Expired" status.

See Also:

- Constant Field Values

STATUS_IN_PROGRESS

`public static final String STATUS_IN_PROGRESS`

The job step "In Progress" status.

See Also:

- Constant Field Values

STATUS_INVOKED

`public static final String STATUS_INVOKED`

The job step "Invoked" status.

See Also:

- Constant Field Values

STATUS_PENDING

`public static final String STATUS_PENDING`

The job step "Pending" status.

See Also:

- Constant Field Values

STATUS_READY

`public static final String STATUS_READY`

The job step "Ready" status.

See Also:

- Constant Field Values

id

`public final Long id`

The job action record (PK).

jobAtionKey

`public final String jobAtionKey`

The job action key.

name

`public final String name`

The job action name.

type

`public final String type`

The job action type.

status

`public final String status`

The job action status.

actionAttempts

`public final Integer actionAttempts`

The job action attempts.

actionMaxAttempts

`public final Integer actionMaxAttempts`

The job action maximum attempts.

actionMessage

`public final String actionMessage`

The job action message.

routeResult

`public final String routeResult`

The job action route result.

externalReferenceNumber

`public final String externalReferenceNumber`

The job action external reference number.

sequence

`public final Integer sequence`

The job action sequence.

assignRepeatIndex

`public final Integer assignRepeatIndex`

The assign repeat index.

assignRepeatItem

`public final String assignRepeatItem`

The assign repeat item.

actionServiceId

`public final Long actionServiceId`

The job action service definition id (PK).

---

txnId

`public final Long txnId`

The action transaction id (PK).

---

timeCreated

`public final Date timeCreated`

The time the job action was created.

---

timeActionExecuted

`public final Date timeActionExecuted`

The time action was last executed.

---

timeActionNextAttempt

`public final Date timeActionNextAttempt`

The time action should next be attempted.

---

timeFinished

`public final Date timeFinished`

The time the job action was finished.

Constructor Detail

JobAction

`public JobAction(com.avoka.fc.core.entity.JobAction jobAction)`

Create a Job Action value object with the given job action entity parameter.

Parameters:

- `jobAction` - the job action entity parameter (required)

JobAction

`public JobAction(Map fields)`

Create a unit testing JobAction value object with the given fields.

Parameters:

- `fields` - the submission entity fields (required)

Since:

- 5.1.4

Method Detail

---

toString

`public String toString()`

Overrides:

- `toString` in class `Object`

Returns:

- a string representation of the object.

JobStep

Package:

- com.avoka.tm.vo

Class JobStep

- java.lang.Object
- com.avoka.tm.vo.JobStep

```
public class JobStep
extends Object
```

Provide a Job Step value object class.

Since:

- 5.0.0

Field Summary

Fields

Modifier and Type	Field and Description
Long	currentJobActionId The current job action id (PK).
Long	expiryServiceId The job step expiry service id (PK).
Long	id The job step record (PK).
boolean	isAllFormsEditable The all forms editable option.
boolean	isDynamicPreConditions The dynamic pre-conditions option.
boolean	isShareExtractData The share form data extracts between step forms option.
boolean	isShareFormData The share form XML data between step forms option.
boolean	isShowPreviousForms The show previous step forms option.
List<JobAction>	jobActions The job step actions.
String	name The job step name.
String	nextStepName The next job step name.
String	status The job step status.
static String	STATUS_CANCELLED The job step "Cancelled" status.
static String	STATUS_COMPLETED The job step "Completed" status.
static String	STATUS_EXPIRED The job step "Expired" status.
static String	STATUS_IN_PROGRESS The job step "In Progress" status.

Date	timeCompletionScheduled The time the job step is scheduled to be completed.
Date	timeCreated The time the job step was created.
Date	timeFinished The time the job step was finished.
String	type The job step type.

Constructor Summary

Constructors

Constructor and Description
JobStep(com.avoka.fc.core.entity.JobStep jobStep) Create a Job Step value object with the given job step entity parameter.
JobStep(Map fields) Create a unit testing JobStep value object with the given fields.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
String	toString()

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

STATUS_CANCELLED

```
public static final String STATUS_CANCELLED
```

The job step "Cancelled" status.

See Also:

- Constant Field Values

STATUS_COMPLETED

```
public static final String STATUS_COMPLETED
```

The job step "Completed" status.

See Also:

- Constant Field Values

STATUS_EXPIRED

```
public static final String STATUS_EXPIRED
```

The job step "Expired" status.

See Also:

- Constant Field Values

STATUS_IN_PROGRESS

```
public static final String STATUS_IN_PROGRESS
```

The job step "In Progress" status.

See Also:

- Constant Field Values

---

id

`public final Long id`

The job step record (PK).

---

name

`public final String name`

The job step name.

---

type

`public final String type`

The job step type.

---

status

`public final String status`

The job step status.

---

currentJobActionId

`public final Long currentJobActionId`

The current job action id (PK).

---

expiryServiceId

`public final Long expiryServiceId`

The job step expiry service id (PK).

---

isAllFormsEditable

`public final boolean isAllFormsEditable`

The all forms editable option.

---

isDynamicPreConditions

`public final boolean isDynamicPreConditions`

The dynamic pre-conditions option.

---

isShareExtractData

`public final boolean isShareExtractData`

The share form data extracts between step forms option.

---

isShareFormData

`public final boolean isShareFormData`

The share form XML data between step forms option.

---

isShowPreviousForms

`public final boolean isShowPreviousForms`

The show previous step forms option.

---

nextStepName

`public final String nextStepName`

The next job step name.

---

timeCreated

`public final Date timeCreated`

The time the job step was created.

timeCompletionScheduled

`public final Date timeCompletionScheduled`

The time the job step is scheduled to be completed.

timeFinished

`public final Date timeFinished`

The time the job step was finished.

jobActions

`public List<JobAction> jobActions`

The job step actions.

Constructor Detail

JobStep

`public JobStep(com.avoka.fc.core.entity.JobStep jobStep)`

Create a Job Step value object with the given job step entity parameter.

Parameters:

- `jobStep` - the job step entity parameter (required)

JobStep

`public JobStep(Map fields)`

Create a unit testing JobStep value object with the given fields.

Parameters:

- `fields` - the submission entity fields (required)

Since:

- 5.1.4

Method Detail

toString

`public String toString()`

Overrides:

- `toString` in class `Object`

Returns:

- a string representation of the object.

Space

Package:

- com.avoka.tm.vo

Class Space

- java.lang.Object
- com.avoka.tm.vo.Space

```
public class Space
extends Object
```

Provide a Form Space value object class.

Since:

- 5.0.0

Field Summary

Fields

Modifier and Type	Field and Description
`String`	`contextPath` The space context path.
`String`	`description` The space name.
`Long`	`id` The space id (PK).
`String`	`name` The space name.
`Map<String,String>`	`propertyMap` The space property name.

Constructor Summary

Constructors

Constructor and Description
`Space(Map fields)` Create a unit testing Space value object with the given fields.
`Space(com.avoka.fc.core.entity.Portal portal)` Create a Space value object with the given portal entity parameter.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`String`	`toString()`

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Field Detail

---

id

```
public final Long id
```

The space id (PK).

---

name

`public final String name`

The space name.

---

description

`public final String description`

The space name.

---

contextPath

`public final String contextPath`

The space context path.

---

propertyMap

`public final Map<String,String> propertyMap`

The space property name.

Constructor Detail

Space

`public Space(com.avoka.fc.core.entity.Portal portal)`

Create a Space value object with the given portal entity parameter.

Parameters:

- `portal` - the portal entity parameter (required)

Space

`public Space(Map fields)`

Create a unit testing Space value object with the given fields.

Parameters:

- `fields` - the submission entity fields (required)

Since:

- 5.1.4

Method Detail

toString

`public String toString()`

Overrides:

- `toString` in class `Object`

Returns:

- a string representation of the object.

SvcConn

Package:

- com.avoka.tm.vo

Class SvcConn

- java.lang.Object
- com.avoka.tm.vo.SvcConn

```
public class SvcConn
extends Object
```

Provides a Service Connection value object.

Since:

- 5.0.0

Field Summary

Fields

Modifier and Type	Field and Description
String	clientCode The organization client code.
String	endpoint The service connection endpoint.
byte[]	fileData The service connection file data.
String	fileName The service connection file name.
Long	id The service connection id (PK).
String	name The service connection name.
Long	orgId The organization client id (PK).
String	param1 The service connection parameter 1.
String	param2 The service connection parameter 2.
String	param3 The service connection parameter 3.
String	param4 The service connection parameter 4.
String	password The service connection password.
String	type The service connection type.
String	username The service connection username.

Constructor Summary

Constructors

Constructor and Description

`SvcConn(Map fields)` Create a unit testing SvcConn value object with the given fields.
`SvcConn(com.avoka.fc.core.entity.ServiceConnection sc)` Create a service connection value object from the given entity parameter.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`String`	`toString()`

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Field Detail

id

`public final Long id`

The service connection id (PK).

name

`public final String name`

The service connection name.

orgId

`public final Long orgId`

The organization client id (PK).

clientCode

`public final String clientCode`

The organization client code.

endpoint

`public final String endpoint`

The service connection endpoint.

type

`public final String type`

The service connection type.

username

`public final String username`

The service connection username.

password

`public final String password`

The service connection password.

param1

`public final String param1`

The service connection parameter 1.

param2

`public final String param2`

The service connection parameter 2.

param3

`public final String param3`

The service connection parameter 3.

param4

`public final String param4`

The service connection parameter 4.

fileData

`public final byte[] fileData`

The service connection file data.

fileName

`public final String fileName`

The service connection file name.

Constructor Detail

SvcConn

`public SvcConn(com.avoka.fc.core.entity.ServiceConnection sc)`

Create a service connection value object from the given entity parameter.

Parameters:

- `sc` - the service connection entity (required)

SvcConn

`public SvcConn(Map fields)`

Create a unit testing SvcConn value object with the given fields.

Parameters:

- `fields` - the submission entity fields (required)

Since:

- 5.1.4

Method Detail

toString

`public String toString()`

Overrides:

- `toString` in class `Object`

Returns:

- a string representation of the object.

SvcDef

Package:

- com.avoka.tm.vo

Class SvcDef

- java.lang.Object
- com.avoka.tm.vo.SvcDef

```
public class SvcDef
extends Object
```

Provide a Service Definition value object class.

Since:

- 5.0.0

Field Summary

Fields

Modifier and Type	Field and Description
String	className The service definition class name.
String	clientCode The organization client code.
Date	createdAt The time the service definition was created.
String	createdBy The user who created the service definition.
String	description The service description.
Long	id The service definition id (PK).
boolean	isCurrentVersion The service is the current active version for the service type.
boolean	isJobTemplate The service definition is a job template.
boolean	isUnitTestEnabled The unit testing is enable for the service.
Date	lastModifiedAt The time the service definition was last modified.
String	lastModifiedBy The user who created the service definition.
String	name The service definition name.
Long	orgId The organization client id (PK).
Map<String,String>	paramsMap The service parameters map.
SvcConn	svcConn The associated service connection.
String	type The service type.

Integer	versionNumber The service definition version number.

Constructor Summary

Constructors

Constructor and Description
SvcDef(Map fields) Create a unit testing SvcDef value object with the given fields.
SvcDef(com.avoka.fc.core.entity.ServiceDefinition sd) Create a service definition value object from the given entity.
SvcDef(SvcDef svcDef, Map<String,String> params) Create a service definition value object from the given entity and put additional parameters to the service definition's parameters map.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
String	toString()

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

id

```
public final Long id
```

The service definition id (PK).

name

```
public final String name
```

The service definition name.

versionNumber

```
public final Integer versionNumber
```

The service definition version number.

orgId

```
public final Long orgId
```

The organization client id (PK).

clientCode

```
public final String clientCode
```

The organization client code.

type

```
public final String type
```

The service type.

description

```
public final String description
```

The service description.

isCurrentVersion

```
public final boolean isCurrentVersion
```

The service is the current active version for the service type.

---

isJobTemplate

```
public final boolean isJobTemplate
```

The service definition is a job template.

---

isUnitTestEnabled

```
public final boolean isUnitTestEnabled
```

The unit testing is enable for the service.

---

className

```
public final String className
```

The service definition class name.

---

createdAt

```
public final Date createdAt
```

The time the service definition was created.

---

createdBy

```
public final String createdBy
```

The user who created the service definition.

---

lastModifiedAt

```
public final Date lastModifiedAt
```

The time the service definition was last modified.

---

lastModifiedBy

```
public final String lastModifiedBy
```

The user who created the service definition.

---

svcConn

```
public final SvcConn svcConn
```

The associated service connection.

---

paramsMap

```
public final Map<String,String> paramsMap
```

The service parameters map.

Constructor Detail

SvcDef

```
public SvcDef(com.avoka.fc.core.entity.ServiceDefinition sd)
```

Create a service definition value object from the given entity.

Parameters:

- `sd` - the service definition entity (required)

SvcDef

```
public SvcDef(SvcDef svcDef,
              Map<String,String> params)
```

Create a service definition value object from the given entity and put additional parameters to the service definition's parameters map.

Parameters:

- `svcDef` - the service definition entity (required)
- `params` - additional parameters to put to service definition parameters (required)

Since:

- 5.0.2

SvcDef

```
public SvcDef(Map fields)
```

Create a unit testing SvcDef value object with the given fields.

Parameters:

- `fields` - the submission entity fields (required)

Since:

- 5.1.4

Method Detail

---

toString

```
public String toString()
```

Overrides:

- `toString` in class `Object`

Returns:

- a string representation of the object.

Txn

Package:

- com.avoka.tm.vo

Class Txn

- java.lang.Object
- com.avoka.tm.vo.Txn

```
public class Txn
extends Object
```

Provide a Transaction value object class.

Since:

- 5.0.0

Field Summary

Fields

Modifier and Type	Field and Description
String	attachmentsStatus The attachment status [Required | Optional | Completed].
List<TxnCheckpoint>	checkpoints The list of transaction delivery checkpoints.
String	clientCode The organization code.
boolean	dataDeleted The transaction user PII data has been deleted.
static String	DELIVERY_COMPLETED The delivery "Completed" status.
static String	DELIVERY_ERROR The delivery "Error" status.
static String	DELIVERY_IN_PROGRESS The delivery "In Progress" status.
static String	DELIVERY_NOT_READY The delivery "Not Ready" status.
static String	DELIVERY_NOT_REQUIRED The delivery "Not Required" status.
static String	DELIVERY_PENDING The delivery "Pending" status.
static String	DELIVERY_READY The delivery "Ready" status.
static String	DELIVERY_SENT_EMAIL The delivery "Sent Email" status.
static String	DELIVERY_UNDELIVERABLE The delivery "Undeliverable" status.
String	deliveryChannel The organization delivery channel name.
Integer	deliveryMaxAttempts The delivery max attempts.
String	deliveryMessage The delivery message.

`String`	`deliveryMethod` The delivery method [Email | Email Secure | Delivery Process | REST Service | Web Service].
`Integer`	`deliveryProcessAttempts` The delivery process attempts.
`String`	`deliveryStatus` The delivery status [Not Ready | Ready | Sent Email | In Progress | Pending | Completed | Error | Undeliverable | Not Required].
`String`	`emailAddress` The contact email address.
`String`	`externalUserId` The transaction external user id.
`List<FileAttach>`	`fileAttachList` The file attachments list.
`static String`	`FORM_ABANDONED` The form "Abandoned" status.
`static String`	`FORM_ASSIGNED` The form "Assigned" status.
`static String`	`FORM_COMPLETED` The form "Completed" status.
`static String`	`FORM_EXPIRED` The form "Expired" status.
`static String`	`FORM_OPENED` The form "Opened" status.
`static String`	`FORM_SAVED` The form "Saved" status.
`static String`	`FORM_SUBMITTED` The form "Submitted" status.
`String`	`formAbandonmentType` The type of form abandonment [Bounced | Started | Cancelled | Submitted | Saved].
`String`	`formCode` The form code, globally unique identifier.
`Map<String,String>`	`formDataMap` The form data map (data extracts).
`Long`	`formId` The form id (PK).
`String`	`formName` The form name.
`String`	`formStatus` The transaction form status [Assigned | Opened | Saved | Submitted | Completed | Expired | Abandoned].
`String`	`formUrl` The form URL if the transaction has not been completed.
`Long`	`formVersionId` The form version id (PK).
`String`	`formVersionNumber` The form version number.
`String`	`formXml` The form XML data.
`Set<String>`	`groupNames` The transaction group names.
`Long`	`id` The transaction id (PK).

Long	jobId The job id.
String	jobRefNumber The job reference number.
Long	orgId The organization id (PK).
String	orgName The organization name.
String	paymentGatewayReceiptNo The payment gateway receipt number.
Date	paymentGatewayTimestamp The payment gateway timestamp.
Long	paymentGatewayTxnNo The payment gateway transaction number.
String	paymentStatus The transaction payment status [Required | Completed | Error | Pending].
Double	paymentTotal The transaction payment total.
String	processStatus The transaction processing status.
Map<String,String>	propertyMap The property map (transaction properties).
String	receiptNumber The transaction receipt number.
byte[]	receiptPdf The receipt PDF bytes.
String	receiptStatus The transaction PDF receipt generation status [Ready | In Progress | Completed | Error | Error No Data].
String	receiptUrl The receipt URL if the transaction has not been completed.
String	saveChallengeHash The save challenge hash
Date	saveChallengeTimeout The save challenge timeout
Long	spaceId The form space id (PK).
String	spaceName The form space name.
String	submitKey The transaction submit key, globally unique id (GUID).
Date	timeAbandoned The time the transaction was abandoned.
Date	timeCompleted The time the transaction was completed.
Date	timeCreated The time the transaction was created.
Date	timeDelivered The time the transaction was delivered.
Date	timeProcessUpdated The time the processing status was last updated.
Date	timeSubmissionExpiry

	The time the submission will expire.
Date	timeSubmitted The time the transaction was submitted.
Date	timeTaskExpiry The time the task will expiry.
Date	timeTaskScheduled The time the task was scheduled to be completed.
Date	timeUserLastModified The time the user last modified the transaction.
String	trackingCode The transaction tracking code.
Integer	txnScore The transaction score.
String	userLoginName The transaction users login name (username).
boolean	userSaved The transaction is user saved.

Constructor Summary

Constructors

Constructor and Description
Txn(Map fields) Create a unit testing Txn value object with the given fields.
Txn(com.avoka.fc.core.entity.Submission sub, Map<String,String> formDataMap, String formXml, Set<String> groupNames, Map<String,String> propertyMap, byte[] receiptPdf, List<FileAttach> fileAttachList) Create a Txn value object with the given parameters.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
String	toString()

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

FORM_ASSIGNED

```
public static final String FORM_ASSIGNED
```

The form "Assigned" status.

See Also:

- Constant Field Values

FORM_OPENED

```
public static final String FORM_OPENED
```

The form "Opened" status.

See Also:

- Constant Field Values

FORM_SAVED

`public static final String FORM_SAVED`

The form "Saved" status.

See Also:

- Constant Field Values

FORM_SUBMITTED

`public static final String FORM_SUBMITTED`

The form "Submitted" status.

See Also:

- Constant Field Values

FORM_COMPLETED

`public static final String FORM_COMPLETED`

The form "Completed" status.

See Also:

- Constant Field Values

FORM_ABANDONED

`public static final String FORM_ABANDONED`

The form "Abandoned" status.

See Also:

- Constant Field Values

FORM_EXPIRED

`public static final String FORM_EXPIRED`

The form "Expired" status.

See Also:

- Constant Field Values

DELIVERY_NOT_READY

`public static final String DELIVERY_NOT_READY`

The delivery "Not Ready" status.

See Also:

- Constant Field Values

DELIVERY_READY

`public static final String DELIVERY_READY`

The delivery "Ready" status.

See Also:

- Constant Field Values

DELIVERY_SENT_EMAIL

`public static final String DELIVERY_SENT_EMAIL`

The delivery "Sent Email" status.

See Also:

- Constant Field Values

DELIVERY_IN_PROGRESS

`public static final String DELIVERY_IN_PROGRESS`

The delivery "In Progress" status.

See Also:

- Constant Field Values

DELIVERY_PENDING

`public static final String DELIVERY_PENDING`

The delivery "Pending" status.

See Also:

- Constant Field Values

DELIVERY_COMPLETED

`public static final String DELIVERY_COMPLETED`

The delivery "Completed" status.

See Also:

- Constant Field Values

DELIVERY_ERROR

`public static final String DELIVERY_ERROR`

The delivery "Error" status.

See Also:

- Constant Field Values

DELIVERY_UNDELIVERABLE

`public static final String DELIVERY_UNDELIVERABLE`

The delivery "Undeliverable" status.

See Also:

- Constant Field Values

DELIVERY_NOT_REQUIRED

`public static final String DELIVERY_NOT_REQUIRED`

The delivery "Not Required" status.

See Also:

- Constant Field Values

---

id

`public final Long id`

The transaction id (PK).

---

attachmentsStatus

`public final String attachmentsStatus`

The attachment status [Required | Optional | Completed].

---

orgId

```
public final Long orgId
```

The organization id (PK).

---

clientCode

```
public final String clientCode
```

The organization code.

---

orgName

```
public final String orgName
```

The organization name.

---

deliveryStatus

```
public final String deliveryStatus
```

The delivery status [Not Ready | Ready | Sent Email | In Progress | Pending | Completed | Error | Undeliverable | Not Required].

---

deliveryChannel

```
public final String deliveryChannel
```

The organization delivery channel name.

---

deliveryMaxAttempts

```
public final Integer deliveryMaxAttempts
```

The delivery max attempts.

---

deliveryMessage

```
public final String deliveryMessage
```

The delivery message.

---

deliveryMethod

```
public final String deliveryMethod
```

The delivery method [Email | Email Secure | Delivery Process | REST Service | Web Service].

---

deliveryProcessAttempts

```
public final Integer deliveryProcessAttempts
```

The delivery process attempts.

---

emailAddress

```
public final String emailAddress
```

The contact email address.

---

formId

```
public final Long formId
```

The form id (PK).

---

formCode

```
public final String formCode
```

The form code, globally unique identifier.

---

formName

```
public final String formName
```

The form name.

---

formStatus

```
public final String formStatus
```

The transaction form status [Assigned | Opened | Saved | Submitted | Completed | Expired | Abandoned].

groupNames

`public final Set<String> groupNames`

The transaction group names.

paymentStatus

`public final String paymentStatus`

The transaction payment status [Required | Completed | Error | Pending].

paymentTotal

`public final Double paymentTotal`

The transaction payment total.

jobId

`public final Long jobId`

The job id.

jobRefNumber

`public final String jobRefNumber`

The job reference number.

spaceId

`public final Long spaceId`

The form space id (PK).

spaceName

`public final String spaceName`

The form space name.

receiptNumber

`public final String receiptNumber`

The transaction receipt number.

receiptStatus

`public final String receiptStatus`

The transaction PDF receipt generation status [Ready | In Progress | Completed | Error | Error No Data].

saveChallengeHash

`public final String saveChallengeHash`

The save challenge hash

Since:

- 5.1.0

saveChallengeTimeout

`public final Date saveChallengeTimeout`

The save challenge timeout

Since:

- 5.1.0

timeCreated

`public final Date timeCreated`

The time the transaction was created.

---

timeAbandoned

`public final Date timeAbandoned`

The time the transaction was abandoned.

---

timeUserLastModified

`public final Date timeUserLastModified`

The time the user last modified the transaction.

---

timeSubmitted

`public final Date timeSubmitted`

The time the transaction was submitted.

---

timeCompleted

`public final Date timeCompleted`

The time the transaction was completed.

---

timeDelivered

`public final Date timeDelivered`

The time the transaction was delivered.

---

timeProcessUpdated

`public final Date timeProcessUpdated`

The time the processing status was last updated.

---

timeSubmissionExpiry

`public final Date timeSubmissionExpiry`

The time the submission will expire.

---

timeTaskExpiry

`public final Date timeTaskExpiry`

The time the task will expiry.

---

timeTaskScheduled

`public final Date timeTaskScheduled`

The time the task was scheduled to be completed.

---

dataDeleted

`public final boolean dataDeleted`

The transaction user PII data has been deleted.

---

submitKey

`public final String submitKey`

The transaction submit key, globally unique id (GUID).

---

trackingCode

`public final String trackingCode`

The transaction tracking code.

---

userLoginName

`public final String userLoginName`

The transaction users login name (username).

---

userSaved

`public final boolean userSaved`

The transaction is user saved.

Since:

- 5.0.3

---

externalUserId

`public final String externalUserId`

The transaction external user id.

---

paymentGatewayTimestamp

`public final Date paymentGatewayTimestamp`

The payment gateway timestamp.

---

paymentGatewayReceiptNo

`public final String paymentGatewayReceiptNo`

The payment gateway receipt number.

---

paymentGatewayTxnNo

`public final Long paymentGatewayTxnNo`

The payment gateway transaction number.

---

processStatus

`public final String processStatus`

The transaction processing status.

---

formVersionId

`public final Long formVersionId`

The form version id (PK).

---

formVersionNumber

`public final String formVersionNumber`

The form version number.

---

txnScore

`public final Integer txnScore`

The transaction score.

---

formAbandonmentType

`public final String formAbandonmentType`

The type of form abandonment [Bounced | Started | Cancelled | Submitted | Saved].

---

formUrl

`public final String formUrl`

The form URL if the transaction has not been completed.

---

receiptUrl

```
public final String receiptUrl
```

The receipt URL if the transaction has not been completed.

checkpoints

```
public final List<TxnCheckpoint> checkpoints
```

The list of transaction delivery checkpoints.

formDataMap

```
public final Map<String,String> formDataMap
```

The form data map (data extracts).

propertyMap

```
public final Map<String,String> propertyMap
```

The property map (transaction properties).

formXml

```
public final String formXml
```

The form XML data.

receiptPdf

```
public final byte[] receiptPdf
```

The receipt PDF bytes.

fileAttachList

```
public final List<FileAttach> fileAttachList
```

The file attachments list.

Constructor Detail

Txn

```
public Txn(com.avoka.fc.core.entity.Submission sub,
           Map<String,String> formDataMap,
           String formXml,
           Set<String> groupNames,
           Map<String,String> propertyMap,
           byte[] receiptPdf,
           List<FileAttach> fileAttachList)
```

Create a Txn value object with the given parameters.

Parameters:

- `sub` - the submission entity (required)
- `formDataMap` - the form data map
- `formXml` - the form XML data
- `groupNames` - the submission group names
- `propertyMap` - the submission property
- `receiptPdf` - the receipt PDF bytes
- `fileAttachList` - the file attachments list

Txn

```
public Txn(Map fields)
```

Create a unit testing Txn value object with the given fields.

Parameters:

- `fields` - the submission entity fields (required)

Since:

- 5.1.4

Method Detail

toString

```
public String toString()
```

Overrides:

- `toString` in class `Object`

Returns:

- a string representation of the object.

TxnCheckpoint

Package:

- com.avoka.tm.vo

Class TxnCheckpoint

- java.lang.Object
- com.avoka.tm.vo.TxnCheckpoint

```
public class TxnCheckpoint
extends Object
```

Provides a Transaction Delivery Checkpoint value object.

Since:

- 5.0.0

Field Summary

Fields

Modifier and Type	Field and Description
String	description The transaction delivery checkpoint description.
String	name The transaction delivery checkpoint name.
String	status The transaction delivery checkpoint status [Registered | Completed | Error].
Date	time The transaction delivery checkpoint time.

Constructor Summary

Constructors

Constructor and Description
TxnCheckpoint(String name, String description, String status, Date time) Create a transaction delivery checkpoint value object.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
String	toString()

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

---

name

```
public final String name
```

The transaction delivery checkpoint name.

---

description

```
public final String description
```

The transaction delivery checkpoint description.

---

status

```
public final String status
```

The transaction delivery checkpoint status [Registered | Completed | Error].

---

time

```
public final Date time
```

The transaction delivery checkpoint time.

Constructor Detail

TxnCheckpoint

```
public TxnCheckpoint(String name,
                     String description,
                     String status,
                     Date time)
```

Create a transaction delivery checkpoint value object.

Parameters:

- `name` - the checkpoint name
- `description` - the checkpoint description
- `status` - the checkpoint status
- `time` - the checkpoint time

Method Detail

---

toString

```
public String toString()
```

Overrides:

- `toString` in class `Object`

Returns:

- a string representation of the object.

User

Package:

- com.avoka.tm.vo

Class User

- java.lang.Object
- com.avoka.tm.vo.User

```
public class User
extends Object
```

Provide a User value object class.

Since:

- 5.0.0

Field Summary

Fields

Modifier and Type	Field and Description
String	accountStatus The user account status [Active | Inactive | Locked | Locked Temporarily | Pending | Rejected].
Date	createdTime The time the user was created.
String	email The user email address.
int	failedLoginAttempts The number of failed login attempts.
String	firstName The user first name (given name).
Set<String>	groupNames The user group names.
boolean	hasGlobalAccess The user has global client organization access.
Long	id The user id (PK).
Date	lastAccessedTime The time the user last logged in.
String	lastName The user last name (family name).
Date	lockoutEndTime The time a temporary lockout should end.
String	loginName The user login name.
String	mobile The user mobile telephone number.
Set<String>	orgNames The user organization names.
Map<String,String>	profileMap The user profile values map.
Set<String>	roleNames The user role names.

`Set<String>`	`spaceNames` The user form space names.
`static String`	`STATUS_ACTIVE` The "Active" user status.
`static String`	`STATUS_INACTIVE` The "Inactive" user status.
`static String`	`STATUS_LOCKED` The "Locked" user status.
`static String`	`STATUS_LOCKED_TEMPORARILY` The "Locked Temporarily" user status.
`static String`	`STATUS_PENDING` The "Pending" user status.
`static String`	`STATUS_REJECTED` The "Rejected" user status.
`static String`	`TYPE_LDAP` The "LDAP" user type.
`static String`	`TYPE_LOCAL` The "Local" user type.
`static String`	`TYPE_SSO` The "SSO" user type.
`String`	`userKey` The user GUID alternative key.
`String`	`userType` The user type [Local | LDAP | SSO].

Constructor Summary

Constructors

Constructor and Description
`User(Map fields)` Create a unit testing User value object with the given fields.
`User(com.avoka.fc.core.entity.UserAccount userAccount)` Create a User value object with the given user account entity.
`User(com.avoka.fc.core.entity.UserAccount userAccount, Set<String> groupNames)` Create a User value object with the given user account entity and group names.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`String`	`toString()`

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Field Detail

STATUS_ACTIVE

```
public static final String STATUS_ACTIVE
```

The "Active" user status.

See Also:

- Constant Field Values

STATUS_INACTIVE

`public static final String STATUS_INACTIVE`

The "Inactive" user status.

See Also:

- Constant Field Values

STATUS_LOCKED

`public static final String STATUS_LOCKED`

The "Locked" user status.

See Also:

- Constant Field Values

STATUS_LOCKED_TEMPORARILY

`public static final String STATUS_LOCKED_TEMPORARILY`

The "Locked Temporarily" user status.

See Also:

- Constant Field Values

STATUS_PENDING

`public static final String STATUS_PENDING`

The "Pending" user status.

See Also:

- Constant Field Values

STATUS_REJECTED

`public static final String STATUS_REJECTED`

The "Rejected" user status.

See Also:

- Constant Field Values

TYPE_LOCAL

`public static final String TYPE_LOCAL`

The "Local" user type.

See Also:

- Constant Field Values

TYPE_LDAP

`public static final String TYPE_LDAP`

The "LDAP" user type.

See Also:

- Constant Field Values

TYPE_SSO

`public static final String TYPE_SSO`

The "SSO" user type.

See Also:

- Constant Field Values

---

id

`public final Long id`

The user id (PK).

---

loginName

`public final String loginName`

The user login name.

---

accountStatus

`public final String accountStatus`

The user account status [Active | Inactive | Locked | Locked Temporarily | Pending | Rejected].

---

email

`public final String email`

The user email address.

---

firstName

`public final String firstName`

The user first name (given name).

---

lastName

`public final String lastName`

The user last name (family name).

---

mobile

`public final String mobile`

The user mobile telephone number.

---

userKey

`public final String userKey`

The user GUID alternative key.

---

userType

`public final String userType`

The user type [Local | LDAP | SSO].

---

hasGlobalAccess

`public final boolean hasGlobalAccess`

The user has global client organization access.

---

failedLoginAttempts

`public final int failedLoginAttempts`

The number of failed login attempts.

---

createdTime

`public final Date createdTime`

The time the user was created.

---

lastAccessedTime

```
public final Date lastAccessedTime
```

The time the user last logged in.

---

lockoutEndTime

```
public final Date lockoutEndTime
```

The time a temporary lockout should end.

---

profileMap

```
public final Map<String,String> profileMap
```

The user profile values map.

---

groupNames

```
public final Set<String> groupNames
```

The user group names.

---

orgNames

```
public final Set<String> orgNames
```

The user organization names.

---

roleNames

```
public final Set<String> roleNames
```

The user role names.

---

spaceNames

```
public final Set<String> spaceNames
```

The user form space names.

Constructor Detail

User

```
public User(com.avoka.fc.core.entity.UserAccount userAccount)
```

Create a User value object with the given user account entity.

Parameters:

- `userAccount` - the user account entity (required)

User

```
public User(com.avoka.fc.core.entity.UserAccount userAccount,
            Set<String> groupNames)
```

Create a User value object with the given user account entity and group names.

Parameters:

- `userAccount` - the user account entity (required)
- `groupNames` - the user group names (required)

User

```
public User(Map fields)
```

Create a unit testing User value object with the given fields.

Parameters:

- `fields` - the submission entity fields (required)

Since:

- 5.1.4

Method Detail

toString

`public String toString()`

Overrides:

- `toString` in class `Object`

Returns:

- a string representation of the object.

Constants

Constant Field Values

Contents

com.avoka.*

com.avoka.tm.test.MockRequest

Modifier and Type	Constant Field	Value
public static final String	REMOTE_USER	"REMOTE_USER"

com.avoka.tm.vo.Job

Modifier and Type	Constant Field	Value
public static final String	STATUS_CANCELLED	"Cancelled"
public static final String	STATUS_COMPLETED	"Completed"
public static final String	STATUS_ERROR	"Error"
public static final String	STATUS_EXPIRED	"Expired"
public static final String	STATUS_IN_PROGRESS	"In Progress"
public static final String	STATUS_PAUSED	"Paused"

com.avoka.tm.vo.JobAction

Modifier and Type	Constant Field	Value
public static final String	STATUS_ASSIGNED	"Assigned"
public static final String	STATUS_CANCELLED	"Cancelled"
public static final String	STATUS_COMPLETED	"Completed"
public static final String	STATUS_ERROR	"Error"
public static final String	STATUS_EXPIRED	"Expired"
public static final String	STATUS_IN_PROGRESS	"In Progress"
public static final String	STATUS_INVOKED	"Invoked"
public static final String	STATUS_PENDING	"Pending"
public static final String	STATUS_READY	"Ready"

com.avoka.tm.vo.JobStep

Modifier and Type	Constant Field	Value
public static final String	STATUS_CANCELLED	"Cancelled"
public static final String	STATUS_COMPLETED	"Completed"
public static final String	STATUS_EXPIRED	"Expired"
public static final String	STATUS_IN_PROGRESS	"In Progress"

com.avoka.tm.vo.Txn

Modifier and Type	Constant Field	Value
public static final String	DELIVERY_COMPLETED	"Completed"
public static final String	DELIVERY_ERROR	"Error"
public static final String	DELIVERY_IN_PROGRESS	"In Progress"
public static final String	DELIVERY_NOT_READY	"Not Ready"
public static final String	DELIVERY_NOT_REQUIRED	"Not Required"
public static final String	DELIVERY_PENDING	"Pending"
public static final String	DELIVERY_READY	"Ready"
public static final String	DELIVERY_SENT_EMAIL	"Sent Email"
public static final String	DELIVERY_UNDELIVERABLE	"Undeliverable"
public static final String	FORM_ABANDONED	"Abandoned"
public static final String	FORM_ASSIGNED	"Assigned"
public static final String	FORM_COMPLETED	"Completed"

public static final String	FORM_EXPIRED	"Expired"
public static final String	FORM_OPENED	"Opened"
public static final String	FORM_SAVED	"Saved"
public static final String	FORM_SUBMITTED	"Submitted"

com.avoka.tm.vo.User

Modifier and Type	Constant Field	Value
public static final String	STATUS_ACTIVE	"Active"
public static final String	STATUS_INACTIVE	"Inactive"
public static final String	STATUS_LOCKED	"Locked"
public static final String	STATUS_LOCKED_TEMPORARILY	"Locked Temporarily"
public static final String	STATUS_PENDING	"Pending"
public static final String	STATUS_REJECTED	"Rejected"
public static final String	TYPE_LDAP	"LDAP"
public static final String	TYPE_LOCAL	"Local"
public static final String	TYPE_SSO	"SSO"

Transact Service Types

Avoka Transact provides a very comprehensive set of service extension points where you plug-in Groovy script to create your customized transaction applications. For an overview of these service extension points please see the Service Types in the Transaction Lifecycle.

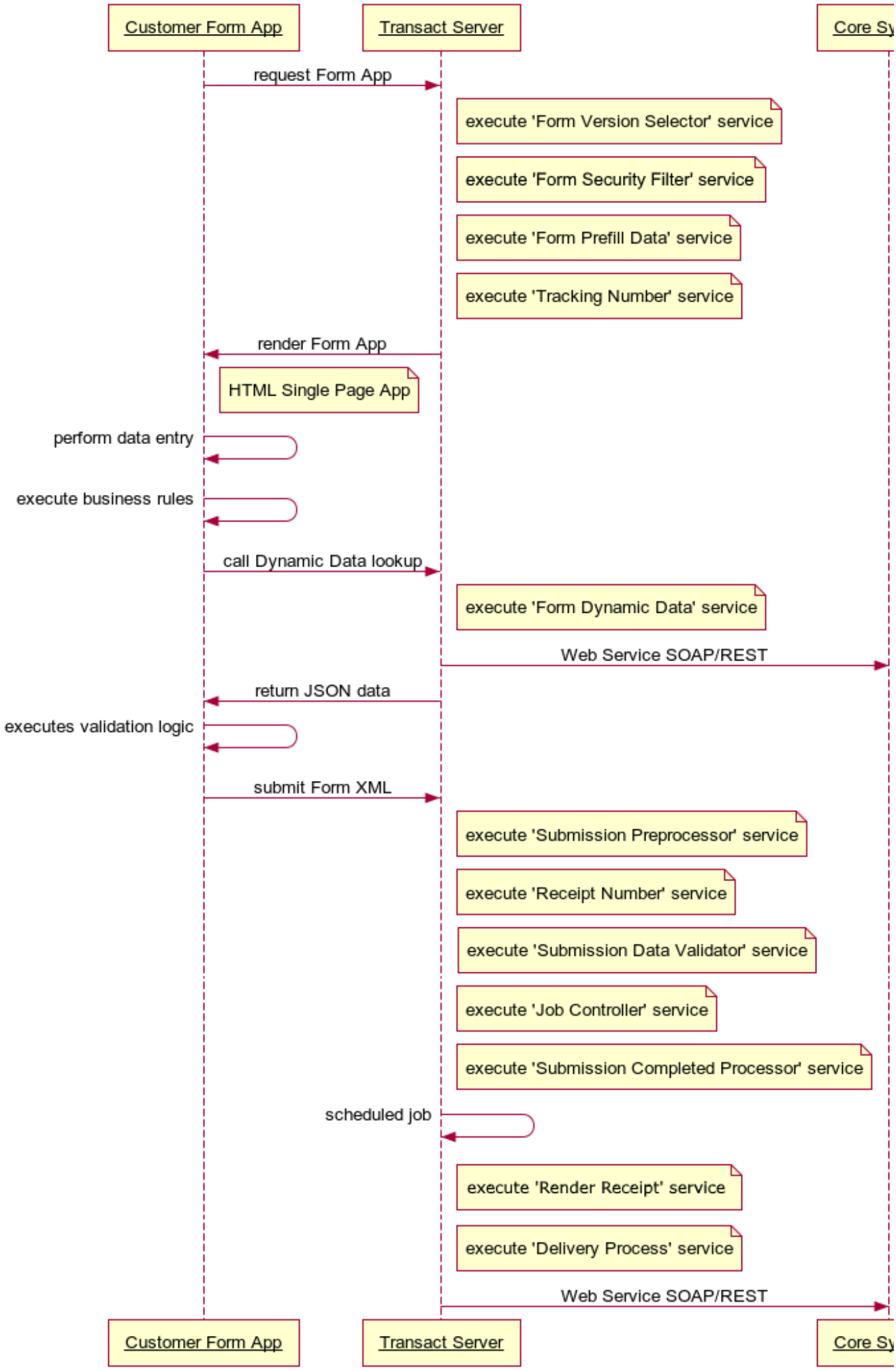
This topic provides Groovy Service API documentation for each of the service types and example Groovy script code to get you started.

Service Types

- Form Version Selector which is used to select which form version to render
- Form Security Filter which is used to perform access checks on when forms are rendered
- Form Prefill Data which is used to pre-populate forms when they are rendered
- Tracking Number which can be used to create custom transaction tracking numbers
- Form Dynamic Data which enable forms to call remote service dynamically based on user selections
- Submission Preprocessor which can validate submitted data before a transaction is created
- Receipt Number which can be used to create custom transaction receipt/reference numbers
- Form Saved Processor which is called when a user saves a form
- Submission Data Validator which is called when a user submits a completed form
- Submission Completed Processor which is called when all the required user steps have been completed
- Render Receipt which is called to render the form submission PDF receipt document
- Delivery Process which is used to deliver the form submission data to back office systems
- Task Expiry Process a service which is called when an assigned task reaches expiry without being completed
- Email Service which is called when an transaction related email is sent to an end user
- Job Action which can be used to provide custom Job action processing
- Scheduled Service an arbitrary service which can be scheduled to execute based on a timer or cron expression
- Groovy Service an arbitrary Groovy service function which can be called from other services
- SSO Revalidation Script called to look at changes which may trigger the SSO filter to re authenticate the user.
- Get SSO Auth Token Script used to obtain an SSO authentication token during user authentication
- Auth OK Response Script called when a user has been successfully authenticated during SSO login
- Groovy Authentication Provider used to authenticate a login token during SSO login
- Transaction History Publisher is used to publish transaction history data to customer's data warehouse
- Virus Scan used to perform the virus scan

Service Types in the Transaction Lifecycle

The sequence diagram below illustrates key service extension points available during transaction processing.


Customer Form App
Transact Server
Core Sy
request Form App
execute 'Form Version Selector' service
execute 'Form Security Filter' service
execute 'Form Prefill Data' service
execute 'Tracking Number' service
render Form App
HTML Single Page App
perform data entry
execute business rules
call Dynamic Data lookup
execute 'Form Dynamic Data' service
Web Service SOAP/REST
return JSON data
executes validation logic
submit Form XML
execute 'Submission Preprocessor' service
execute 'Receipt Number' service
execute 'Submission Data Validator' service
execute 'Job Controller' service
execute 'Submission Completed Processor' service
scheduled job
execute 'Render Receipt' service
execute 'Delivery Process' service
Web Service SOAP/REST
Customer Form App
Transact Server
Core Sy

Delivery Process

The Delivery Process service is used to deliver data associated with a completed submission, including submitted form XML, file attachments and the PDF receipt document to external systems.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
txn	the transaction record value object, see Txn	no

Error Handling

Submission delivery is performed asynchronously by a background job. If a delivery process error occurs the end user will never see this error.

If an exception is thrown by the Delivery Process, the **`SubmissionDeliveryController`** will make the submission delivery status **`Error`** and will log the error in the TM database error log.

Please note the SubmissionDeliveryController will rollback the current delivery transaction if an unhandled error is thrown by the Delivery Process. Any TM database changed performed by the Groovy Script will be rolled back with the transaction, including any delivery checkpoint changes.

If the Service Definition is configured to perform a number of automatic retry attempts, then the submission will be scheduled to retry delivery after the configured period of time.

Service Template

This section shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

class FluentDeliveryProcess {

    /*
     * Perform Transation Delivery Process
     *
     * returns: null if completed or DeliveryResult objet
     */
    DeliveryResult invoke(SvcDef svcDef, Txn txn) {

        try {
            // TODO: perform delivery work

            return new DeliveryResultBuilder()
                .setStatus(Txn.DELIVERY_COMPLETED)
                .build()

        } catch (Exception e) {
            return new DeliveryResultBuilder()
                .setMaxDeliveryAttempts(5)
                .setMessage(e)
                .setNextDeliveryMins(30)
                .setStatus(Txn.DELIVERY_ERROR)
                .build()
        }
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
```

```
     *
     * throws: exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        String xmlData = testParams['Test XML Data']
        Txn txn = new MockVoBuilder().createTxnDeliveryReadyWithXml(xmlData)

        Map params = [
            "svcDef": svcDef,
            "txn": txn
        ]

        DeliveryResult result = (DeliveryResult) new ServiceInvoker(svcDef).invoke(params)

        // Check result
        logger.info result

        assert result.status == Txn.DELIVERY_COMPLETED
    }
}
```

Email Service

The Email Service is called whenever a notification needs to be sent to a TM user. TM provides email-based implementations and allows you to implement your own services as well (which do not necessarily have to use email, for example you could provide an SMS notification service).

The Email Service can be configured on multiple levels and is resolved in the following order:

1. Form Version
2. Organization
3. Global Service

The global default Email Service is used throughout, but can be overridden on the organization and form version level for sending submission related messages to applicants.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
txn	the transaction record value object, see Txn	yes
subject	the email subject	no
message	the email body message	yes
fromAddress	the sender address	yes
replyToAddress	the address that shall be used when the recipient replies to the message	yes
toAddress	the recipient address(es)	no
ccAddress	the address(es) to send copies of the email to	yes
bccAddress	the address(es) to send blind carbon copies of the email to	yes
attachmentMap	map of attached files (containing filename/file data entries)	yes

Error Handling

If an error occurs an error will be recorded in the TM database error log.

Service Template

This section shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentEmailService {

    /**
     * Perform send email
     */
    void invoke(SvcDef svcDef,
                Txn txn,
                String subject,
                String message,
                String fromAddress,
                String replyToAddress,
                String toAddress,
                String ccAddress,
                String bccAddress,
                Map<String, byte[]>attachmentMap) {

        // TODO: perform business logic

        new Emailer()
            .setSubject(subject)
            .setMessage(message)
            .setFromAddress(fromAddress)
            .setReplyToAddress(replyToAddress)
            .setToAddress(toAddress)
            .setCcAddress(ccAddress)
            .setBccAddress(bccAddress)
            .setAttachmentMap(attachmentMap)
            .setTxn(txn)
```

```
            .sendEmail()
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        Txn txn = new MockVoBuilder().createTxnOpened()

        Map params = [
            "svcDef": svcDef,
            "txn": txn,
            "subject": "Email Subject",
            "message": "Email Message",
            "fromAddress": null,
            "replyToAddress": null,
            "toAddress": "test@avoka.com",
            "ccAddress": null,
            "bccAddress": null,
            "attachmentMap": null,
        ]

        new ServiceInvoker(svcDef).invoke(params)
    }
}
```

Form Dynamic Data

The Form Dynamic Data service is used provide forms with dynamic data in real-time while the user is interacting with the form.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
txn	the transaction record value object, see Txn	yes
request	the servlet HttpServletRequest	no
user	the authenticated user, see User	yes

Script Result

The script should return a JSON string value to be bound into the HTML form's data model.

Error Handling

If your Groovy script throws an error the **`FormDynamicDataServlet`** will return a HTTP 500 error code back to the form. It is up to the form to handle the error returned. The system will also log the error to the TM database event log specifying an event type of 'Error'.

Service Template

This section shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentDynamicData {

    /*
     * Perform Form Dynamic Data service call.
     *
     * returns: REST response text data
     */
    String invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {

        // TODO: replace with data lookup call

        String data = '''{
          "address": {
            "firstLine": "123 Wall Street"
          }
        }'''

        return data
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws: exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        Txn txn = new MockVoBuilder().createTxnOpened()
```

```
        MockRequest request = new MockRequest()

        Map params = [
            "svcDef": svcDef,
            "txn": txn,
            "request": request,
            "user": null
        ]

        String result = (String) new ServiceInvoker(svcDef).invoke(params)

        logger.info result

        Path path = new Path(result)

        assert "123 Wall Street" == path.val("address.firstLine")
    }

}
```

Form Prefill Data

Form Prefill

The Form Prefill service is used provide XML pre-population data to the form when it is first rendered. Please note this service is not called when someone subsequently opens a saved form.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
form	the form value object, see Form	no
formXml	the XML Document object which is used to pre-fill the form with	no
txn	the transaction record value object, see Txn	yes
request	the servlet HttpServletRequest	yes
user	the authenticated user, see User	yes

Script Result

The script can either return a form XML string value or update the schemaSeed XML Document parameter object. If the script returns an XML data then this value will be either:

- used as the form's XML data Document, when no Input XML Prefill Mapping are configured
- used as input XML prefill data which will be mapped into the forms XML data Document, if Input XML Prefill Mapping are configured

If the script does not return any XML or update the schemaSeed XML Document parameter, then the form will use its default prefill data instead.

Error Handling

If your Groovy script throws an error when executing the exception will be logged to the TM database error log and the user will be redirected to a form error page which will contain a Error Reference Number. This reference number is the TM database error log ID which you can use to look up the error. The error log record will contain useful contextual information about the error.

If, at prefill time, you determine you don't want the user to view the form you can redirect away to another page using a RedirectException.

```
import com.avoka.tm.util.RedirectException

throw new RedirectException('/form-expired.htm?formCode=' + form.formCode)
```

Please note in TransactField App you cannot use this design pattern, as the Form Prefill Service is executed on the server and not in the client.

Service Template

This section shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import javax.servlet.http.*
import org.w3c.dom.*

class FluentFormPrefill {

    /*
     * Perform form prefill service
     *
     * return: the form XML prefill document
     * throws: RedirectException to redirect user to another page
     */
    Document invoke(SvcDef svcDef, Form form, Document formXml, Txn txn, HttpServletRequest request, User user)
throws RedirectException {

        // TODO: replace with data lookup call

        XmlDoc xmlDoc = new XmlDoc(formXml)

        xmlDoc.setText("/AvokaSmartForm/Customer/Address/PostCode", "9999")
```

```
        return xmlDoc.document
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import org.w3c.dom.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        Document formXml = new XmlDoc(testParams['Test XML Data']).document
        MockRequest request = new MockRequest()

        Map params = [
            "svcDef": svcDef,
            "form": null,
            "formXml": formXml,
            "txn": null,
            "request": request,
            "user": null
        ]

        Document result = (Document) new ServiceInvoker(svcDef).invoke(params)

        assert result != null

        Path path = new Path(result)

        assert "9999" == path.val("//Customer/Address/PostCode")
    }
}
```

Form Saved Processor

The Form Saved Processor is called immediately after the user has saved a form to TM.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
txn	the transaction record value object, see Txn	no
request	the servlet HttpServletRequest	yes
user	the authenticated user, see User	yes

Error Handling

If an error occurs the system will catch the error and log the error to the TM database error log. The user should not be aware that any error has occurred.

Please note any changes made by the service made to the database will be persisted with the transaction, and it will be up to the service to undo any changes it has made.

Service Template

This section shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentFormSavedProcessor {

    /**
     * Perform form save processor service
     */
    void invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {

        // TODO: perform business logic

        new TxnUpdater(txn)
            .setProperty("IDV-Status", "Pending")
            .update()
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.query.*
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        String xmlData = testParams['Test XML Data']
        Txn txn = new MockVoBuilder().createTxnSavedWithXml(xmlData)
        MockRequest request = new MockRequest()

        Map params = [
            "svcDef": svcDef,
            "txn": txn,
            "request": request,
```

```
            "user": null
        ]

        new ServiceInvoker(svcDef).invoke(params)
        txn = new TxnQuery()
            .setTxn(txn)
            .withPropertyMap()
            .firstValue()

        logger.info txn

        assert txn.formStatus == Txn.FORM_SAVED

        assert txn.propertyMap["IDV-Status"] == 'Pending'
    }
}
```

Form Security Filter

The form security filter service is used to perform additional security and access control checks when someone renders a form or reopens a saved transaction. The form security filter service is called before TM access checks run.

This service is configured via the form version "Services" tab.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
form	the form value object, see Form	no
txn	the transaction record value object, see Txn	yes
request	the servlet HttpServletRequest	no
user	the authenticated user, see User	yes

Script Result

If all checks performed by the script pass, the script should do nothing. TM will continue with in-built access checks before allowing the user to access the form or transaction.

If any checks fail and access shall be denied to the user, the script must throw a RedirectException with a valid target attribute. TM will redirect the user to the target.

Please note in TransactField App you cannot use this design pattern, as the form security filter service is executed on the server and not in the client.

Service Template

This section shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentFormSecurityFilter {

    /*
     * Perform form security filter service
     *
     * throws: if access checks fail, throw a RedirectException to redirect to another page
     */
    Object invoke(SvcDef svcDef, Form form, Txn txn, HttpServletRequest request, User user) throws
RedirectException {

        // Opening a new form
        if (txn == null) {
            return
        }

        // TODO: replace this with realistic access control checks
        def authorization = request.getHeader("Authorization")

        if (authorization == null) {
            // access control checks failed, redirect to an appropriate URL
            throw new RedirectException("../not-authorized.htm")
        }

        // Store the security tokens in session
        Security.addSessionTxnSecurityTokens(request, txn)
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
```

```
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws: exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        MockRequest request = new MockRequest()

        // Test 1: open form flow
        Map params = [
            "svcDef": svcDef,
            "form": null,
            "txn": null,
            "request": request,
            "user": null
        ]

        try {
            new ServiceInvoker(svcDef).invoke(params)
        } catch (Exception re) {
            assert false : 'Test 1 failed'
        }

        // Test 2: we ensure that the security filter does not grant access for unauthorized requests
        Txn txn = new MockVoBuilder().createTxnCompletedWithXml("<AvokaSmartForm/>")
        params.txn = txn

        try {
            new ServiceInvoker(svcDef).invoke(params)
            // the above call should fail due to access attribute not having been set up
            assert false : 'Test 2 failed'
        } catch (RedirectException re) {
            assert "../not-authorized.htm" == re.getTarget()
        }

        // Test 3: we set up the appropriate header attribute that will cause access control checks to pass
        request.setHeader("Authorization", "3888c972a167ca55e967cd764ab691bf")

        try {
            new ServiceInvoker(svcDef).invoke(params)
        } catch (Exception re) {
            logger.info re
            assert false : 'Test 3 failed'
        }
    }
}
```

Form Version Selector

The Form Version Selector service is used to select the version of a form to render when a user starts accessing the form.
For simple purposes, the current version of a form can be used; or for example in A/B testing, a version of the form may be selected either completely at random or depending on certain characteristics of the user.

Please see the Avoka Transact Services topic Form Version Selector.

This service is configured via the form version "Services" tab.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
form	the form value object, see Form	no
request	the servlet HttpServletRequest	no
user	the authenticated user, see User	yes

Script Result

The script can return String object:

- if the returned object is null, then the current template version will be resolved,
- otherwise the returned String object will be resolved to the selected template version.

Examples

Region Form Version Selection

The example below chooses a region form version.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentFormVersionSelector {

    /*
     * Perform form version selector service
     */
    String invoke(SvcDef svcDef, Form form, HttpServletRequest request, User user) {

        String country = request.getParameter('country')

        if ('AU'.equals(country)) {
            return '1.0'
        }

        if ('DE'.equals(country)) {
            return '2.0'
        }

        if ('JP'.equals(country)) {
            return '3.0'
        }

        return null
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.vo.*

class UnitTest {
```

```
    /*
     * Perform service unit test
     *
     * throws: exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        Form form = new MockVoBuilder().createForm()
        MockRequest request = new MockRequest()
        request.setParameter('country','AU')

        Map params = [
            "svcDef": svcDef,
            "form": form,
            "request": request,
            "user": null
        ]

        try {
            String result = new ServiceInvoker(svcDef).invoke(params)

            assert result == '1.0'

        } catch (Exception re) {
            assert false : 'Test failed'
        }
    }
}
```

Groovy Service

Provides a generic Fluent API Groovy script service which can provide reusable scripts to be called by other services.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
request	the servlet HttpServletRequest	no
user	the authenticated user, see User	yes
params	the map of parameters provided by the service caller, or request parameters map if called from REST Groovy Service Invoke API	yes
params.job	associated job action Job, added when Groovy Service called from `$func.invoke()`	yes
params.jobAction	associated JobAction, added when Groovy Service called from `$func.invoke()`	yes

Script Result

The script can optionally return a value.

Error Handling

If an error occurs invoking the Groovy Service then the error will be recorded in the TM database error log.

Service Template

This section shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentGroovyService {

    /*
     * Perform a groovy service invocation
     *
     * return: the result object
     */
    Object invoke(SvcDef svcDef, HttpServletRequest request, User user, Map params) {

        // TODO: perform your logic
        def result = 'groovy result - ' + params.formId

        return result
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        Map args = [
            "formId": 23
        ]
```

```
        Map params = [
            "svcDef": svcDef,
            "request": null,
            "user": null,
            "params": args
        ]

        def result = new ServiceInvoker(svcDef).invoke(params)
        // Check result
        logger.info result

        assert "groovy result - 23" == result
    }
}
```

Job Action

The Job Action service is used to provide a custom Job step action service for job processing. This could be used to perform some custom processing such as calling an external system.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
job	the transaction record value object, see Job	no
jobAction	the transaction record value object, see JobAction	no
request	the servlet HttpServletRequest	no
user	the authenticated user, see User	yes

Error Handling

Job action processing is generally performed asynchronously by a background job. If a error occurs with asynchronous processing the end user will never see this error. If an exception is thrown by the Job Action service, the `JobControllerService` log the error to the will log the error in the TM database error log.

If the Job is configured to process submissions immediately ("processSubmitImmediate": "true"), then the job action may be processed in the user thread depending upon the the job definition. If a job action executing in the user thread throws an exception then the submission transaction will be rolled back, and then an generalized transaction error will will be logged in the TM database error log. Please note the normal job error logging to the database will not be available as the entire submission transaction has been rolled back, to determine the root cause of an error you will probably need to review the server log file.

It is extremely important that any custom actions running in the user thread handle errors gracefully and do not throw an exception. The action should also execute quickly so the user experience is not degraded and the systems performance is not impacted under heavy loads.

Depending upon the services configuration the JobControllerService will perform a number of automatic retry attempts, to process the action service.

Service Template

This section shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.job.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentJobActionService {

    /*
     * Perform a job action service invocation
     *
     * return: the job action result
     */
    ActionResult invoke(SvcDef svcDef, Job job, JobAction jobAction, HttpServletRequest request, User user) {

        // TODO: perform your logic

        String xml = Jobs.getStartTxnXml(job)

        Path path = new Path(xml)

        String email = path.val('//Contact/Email')

        return new ActionResultBuilder()
            .setStatus('Completed')
            .setMessage('Action completed for: ' + email)
            .build()
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.job.*
```

```
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        String formXml = testParams['Test XML Data']
        Job job = new MockVoBuilder().createJobInProgressWithXml(formXml)
        JobAction jobAction = job.jobSteps[0].jobActions[0]
        MockRequest request = new MockRequest()

        Map params = [
            "svcDef": svcDef,
            "job": job,
            "jobAction": jobAction,
            "request": request,
            "user": null
        ]

        ActionResult result = (ActionResult) new ServiceInvoker(svcDef).invoke(params)

        logger.info 'result: ' + result

        assert result != null

        assert result.message == 'Action completed for: john.smith@avoka.com'
    }
}
```

Receipt Number

The Receipt Number service is used to generate receipt or reference numbers for form submission transactions. These receipt numbers are included in the PDF receipt document, and can also be used as a general reference number for the transaction.

Please note by default the transaction's tracking code is displayed as the transact reference number in associated transaction pages and emails.

This service is configured via the Form Details tab.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
form	the transaction record value object, see Form	no
txn	the transaction record value object, see Txn	no
request	the servlet HttpServletRequest	yes
user	the authenticated user, see User	yes

Script Result

The script should return a transaction receipt number.

Error Handling

If the service returns null or a blank value, or throws an exception, the **`FluentReceiptNumberService`** will generate a default value to be used based on the form code and the submission ID, e.g. **`CSA-2948`**. This is to ensure the end user will have a receipt number that they can reference.

Any **`Exception`** thrown by the Groovy script will be logged against the submission record in the TM database, and will also be logged in the server log file. This error handling is to ensure the users submission transaction is not rolled back and lost because of some error in the Groovy script.

Service Template

This section shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentReceiptNumberService {

    /*
     * Perform a transaction receipt number generation
     *
     * return: a transaction receipt number, values longer than 100 characters will be trimmed
     */
    String invoke(SvcDef svcDef, Form form, Txn txn, HttpServletRequest request, User user) {

        return form.formCode + "-" + txn.id
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {
```

```
        String formXml = testParams['Test XML Data']
        MockVoBuilder mob = new MockVoBuilder()
        Form form = mob.createForm()
        Txn txn = mob.createTxnSubmittedWithFormAndXml(form, formXml)
        MockRequest request = new MockRequest()

        Map params = [
            "svcDef": svcDef,
            "form": form,
            "txn": txn,
            "request": request,
            "user": null
        ]

        String receiptNumber = (String) new ServiceInvoker(svcDef).invoke(params)

        logger.info "Receipt Number: " + receiptNumber

        // Test any business logic performed
        assert receiptNumber == (form.formCode + "-" + txn.id)
    }
}
```

Render Receipt

Render Receipt Service

The Render Receipt Service is used to render a receipt pdf file.

Please see the Avoka Transact Services topic Render Receipt Service.

This service is configured via the form version "Services" tab.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
form	the form value object, see Form	no
txn	the transaction record value object, see Txn	no
request	the servlet HttpServletRequest	yes
user	the authenticated user, see User	yes

Script Result

The script can return byte[] object:

- if the returned object is null, then exception will be thrown,
- otherwise the returned byte[] object will rendered to pdf.

Service Template

This section shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentRenderReceiptService {

    /*
     * Perform render receipt service
     */
    byte[] invoke(SvcDef svcDef, Form form, Txn txn, HttpServletRequest request, User user) {

        //TODO: generate pdf data
        byte[] data = "pdf".getBytes()

        return data
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws: exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        MockVoBuilder mob = new MockVoBuilder()
        Form form = mob.createForm()
```

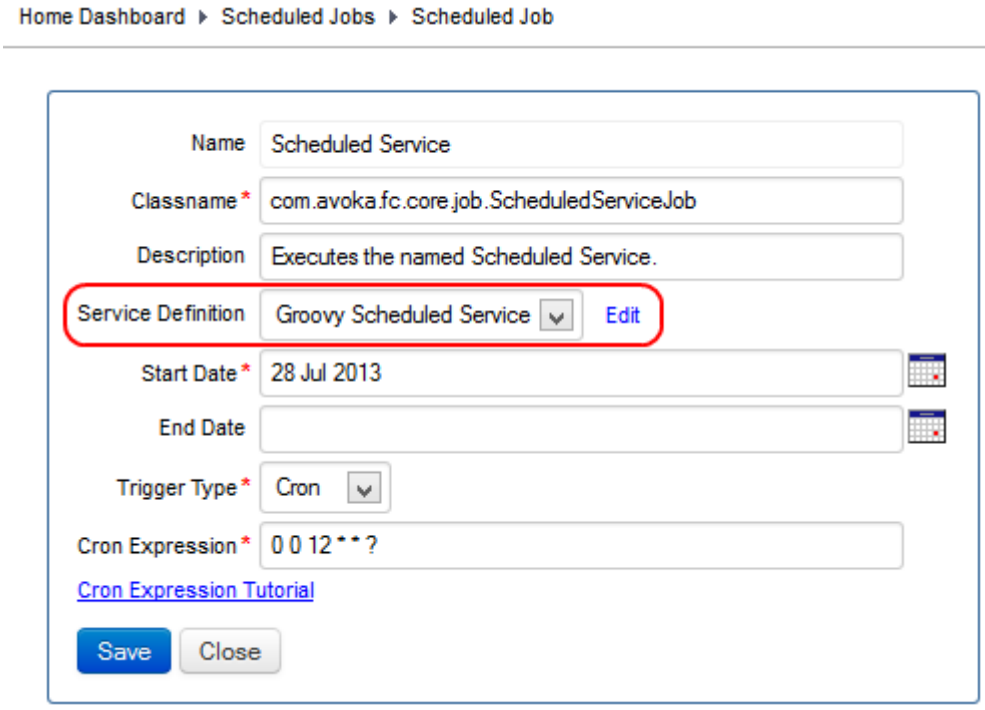
```
        Txn txn = mob.createTxnSubmittedWithFormAndXml(form)
        MockRequest request = new MockRequest()

        Map params = [
            "svcDef": svcDef,
            "form": form,
            "txn": txn,
            "request": request,
            "user": null
        ]

        try {
            byte[] result = (byte[]) new ServiceInvoker(svcDef).invoke(params)

            assert result != null

        } catch (Exception re) {
            assert false : 'Test failed'
        }
    }
}
```

Scheduled Service

The Scheduled Service provides a Groovy service which can be called at a scheduled time using a configured **`ScheduledServiceJob`** instance. This is useful for performing background jobs at regular or scheduled times.

This service is configured via System > Service Definitions.

You will also need to configure a new Scheduled Service job instance to call your named service. Please see the example Scheduled Service job definition below.



Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no

Error Handling

If an error occurs invoking the Scheduled Service then the error will be recorded in the TM database error log.

Service Template

This section shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.vo.*

class FluentScheduledService {

    /*
     * Perform scheduled job
     *
     * returns: string message to be logged to the server log as an INFO message
     */
    String invoke(SvcDef svcDef) {

        // TODO: perform business logic

        return 'Schedule Service invoked'
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.util.*
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        Map params = [:]
        params.svcDef = svcDef

        String result = (String) new ServiceInvoker(svcDef).invoke(params)

        logger.info result

        assert result == 'Schedule Service invoked'
    }
}
```

Submission Complete Processor

The Submission Completed Processor is called immediately after a form submission has progressed to form status 'Completed'. This could occur immediately after submission, or later when the user completes attachments or payment steps.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
txn	the transaction record value object, see Txn	no
request	the servlet HttpServletRequest	yes
user	the authenticated user, see User	yes

Error Handling

If an error occurs the system will catch the error and log the error to the TM database error log. The user should not be aware that any error has occurred.

Please note any changes made by the service made to the database will be persisted with the transaction, and it will be up to the service to undo any changes it has made.

Service Template

This section shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentSubmissionCompletedProcessor {

    /**
     * Perform submission completed processor service
     */
    void invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {

        // TODO: perform business logic

        new TxnUpdater(txn)
            .setProperty("IDV-Status", "Completed")
            .update()

        new EventLogger()
            .setMessage("User '" + user.loginName + "' completed Identity Verification")
            .setRequest(request)
            .setTxn(txn)
            .logInfo()
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

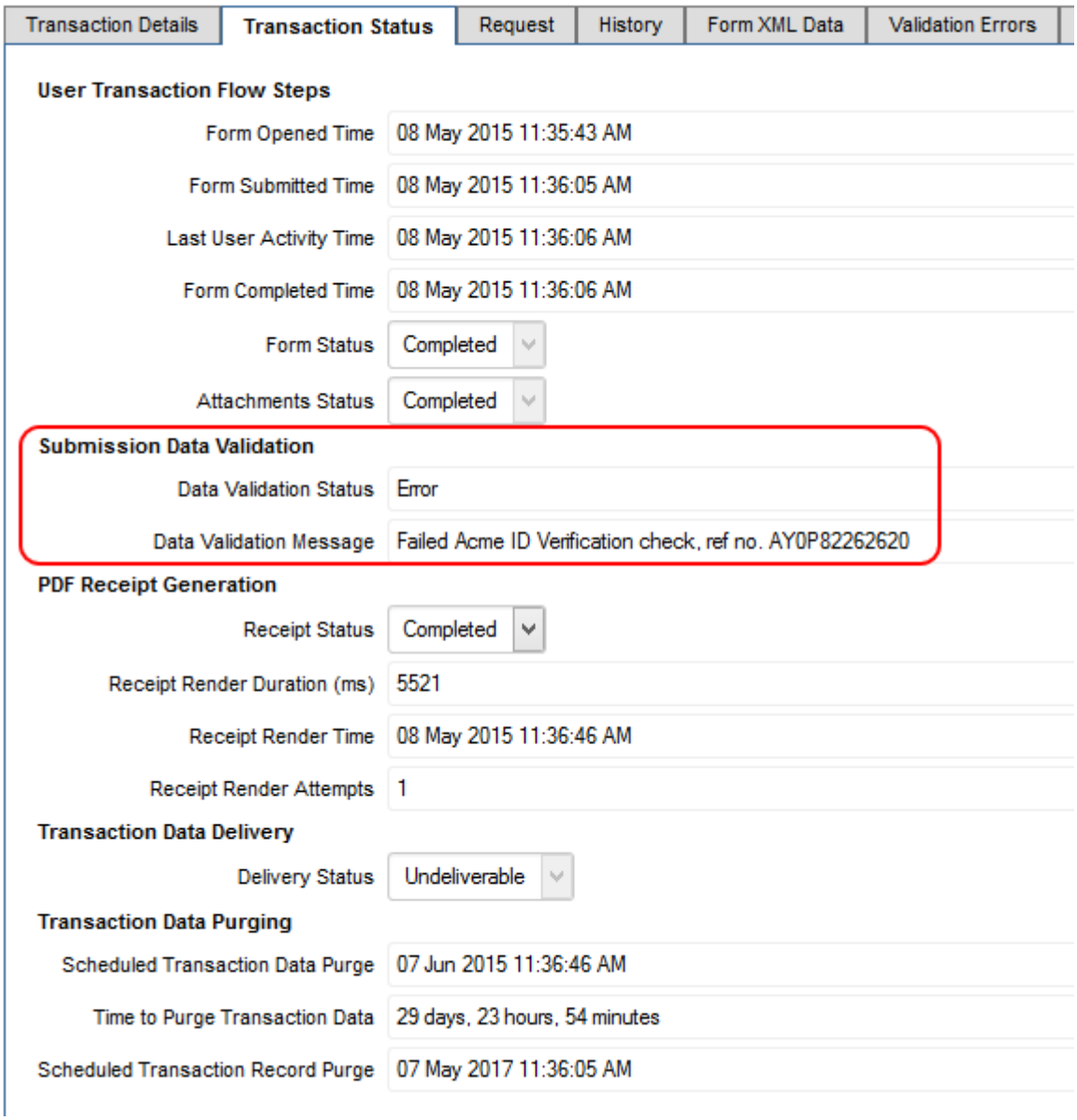
```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.query.*
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        String xmlData = testParams['Test XML Data']
```

```
        Txn txn = new MockVoBuilder().createTxnCompletedWithXml(xmlData)
        MockRequest request = new MockRequest()
        User user = new MockVoBuilder().createUserLocal()

        Map params = [
            "svcDef": svcDef,
            "txn": txn,
            "request": request,
            "user": user
        ]

        new ServiceInvoker(svcDef).invoke(params)

        txn = new TxnQuery()
            .setId(txn.id)
            .withPropertyMap()
            .firstValue()

        logger.info txn

        assert txn.formStatus == Txn.FORM_COMPLETED

        assert txn.propertyMap["IDV-Status"] == 'Completed'
    }
}
```

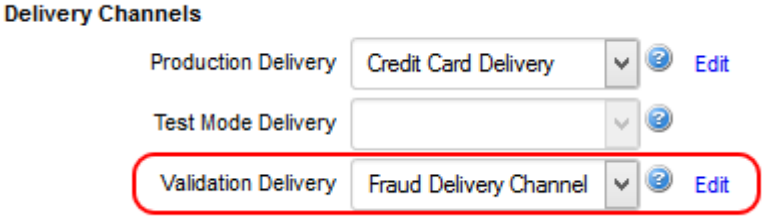
Submission Data Validator

The Submission Data Validator is called when a completed form is submitted. This service can be used to verify that the submitted XML is valid and not an attempted fraudulent transaction.

A Submission Data Validator service may perform business rules validation on the server using Groovy Script interrogating the submitted XML. If the service finds validation errors it can return a list of validation errors which will then be stored against the transaction, and is visible using the 'Validation Errors' tab. If the service returns one or more validation error strings, then the transaction will have its Data Validation status set to 'Error'.



Transactions with a Error data validation status will not be delivered using the normal delivery channel. If a Form Validation Delivery Channel is defined then the transaction will be delivered using this channel, otherwise it will be marked as Undeliverable and made ready for submission data purging.



Please see the Avoka Transact Services topic Submission Data Validator.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no

txn	the transaction record value object, see Txn	no
request	the servlet HttpServletRequest	yes
user	the authenticated user, see User	yes

Error Handling

If an error occurs the system will catch the error and log the error to the TM database error log. The user should not be aware that any error has occurred.

Please note any changes made by the service made to the database will be persisted with the transaction, and it will be up to the service to undo any changes it has made.

Service Template

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentSubmissionDataValidator {

    /*
     * Perform submission data validator service
     *
     * return: list of data validation error messages
     */
    List<String> invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {

        // TODO: perform validation logic, return any data validation errors in error list
        List errors = new ArrayList<String>()

        return errors
    }
}
```

Unit Test Template

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.query.*
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        String xmlData = testParams['Test XML Data']
        Txn txn = new MockVoBuilder().createTxnSubmittedWithXml(xmlData)
        MockRequest request = new MockRequest()

        Map params = [
            "svcDef": svcDef,
            "txn": txn,
            "request": request,
            "user": null
        ]

        List errors = (List) new ServiceInvoker(svcDef).invoke(params)

        txn = new TxnQuery()
            .setId(txn.id)
            .withPropertyMap()
            .firstValue()

        logger.info txn

        // Test any business logic performed
        assert txn.formStatus == Txn.FORM_SUBMITTED
```

```
        assert errors.size() == 0
    }
}
```

Submission Preprocessor

The Submission Preprocessor is called when the form XML data has been saved or submitted (POST) to the server. This service is invoked before the form submission transaction has started, and provides an interceptor point where you can validate the form XML before it is saved.

This service is configured via the Form Version Services tab.

Please Note: this service is not available for TransactField App submissions.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
txn	the transaction record value object, see Txn	yes
formXml	the saved or submitted form XML data	no
request	the servlet HttpServletRequest	no
user	the authenticated user, see User	yes

Script Result

The script should return a submissionDataBean which the submitted XML data.

Error Handling

If your Groovy script throws an error when executing the exception will be logged to the TM database error log and the user will be redirected to a form error page which will contain a Error Reference Number. This reference number is the TM database error log ID which you can use to look up the error. The Error Log record will contain useful contextual information about the error.

If you want to simply send the user away to an arbitrary page without logging an error you can simply throw a RedirectException to the target page.

Service Template

This section shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentSubmissionPreprocessor {

    /*
     * Perform a form save/submission preprocessor
     *
     * return: null if there are no changes to form XML, or a modified form XML string
     * throws: RedirectException to reject submission and redirect user to another page
     */
    String invoke(SvcDef svcDef, String formXml, Txn txn, HttpServletRequest request, User user) throws
RedirectException {

        if (!Security.isXmlTextSafe(formXml)) {
            throw new RedirectException('../not-authorized.htm')
        }

        // If XML valid return null to continue
        return null
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.query.*
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
```

```
class UnitTest {

    /*
     * Perform service unit test
     *
     * throws exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        String formXml = testParams['Test XML Data']
        MockRequest request = new MockRequest()

        Map params = [
            "svcDef": svcDef,
            "formXml": formXml,
            "txn": null,
            "request": request,
            "user": null
        ]

        try {
            def result = new ServiceInvoker(svcDef).invoke(params)
            assert result == null
        } catch (RedirectException re) {
            assert false
        }

        // Test XSS attack data
        params.formXml = "<SmartForm>alert('hi there')</SmartForm>"

        try {
            def result = new ServiceInvoker(svcDef).invoke(params)
            assert false
        } catch (RedirectException re) {
            logger.info re
            assert re.target == '../not-authorized.htm'
        }
    }
}
```

Task Expiry Process

The Task Expiry Process is called by the Transaction Processor job when a task has expired. A task is considered to have expired when it has not been completed by its scheduled expiry date. Expired tasks are removed from users' Task List.

This service is configured via the Form Version Services tab.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
txn	the transaction record value object, see Txn	no

Error Handling

If an error occurs attempting to expire a task the error will be recorded in the TM database error log. A Task Expiry service can have a configurable number of automatic retry attempts. The default configuration is to have 5 attempts.

Service Template

This section shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

class FluentTaskExpiry {

    /*
     * Perform Task expiry process
     */
    void invoke(SvcDef svcDef, Txn txn) {

        // TODO: perform task expiry work

        String msg = txn.formStatus + " task " + txn.trackingCode + " assigned to " + txn.emailAddress + " has
expired"

        new Emailer()
            .setToAddress("task-manager@avoka.com")
            .setSubject("Task " + txn.trackingCode + " - " + txn.formStatus + " has expired")
            .setMessage(msg)
            .setTxn(txn)
            .sendEmail()

        new EventLogger()
            .setMessage(msg)
            .setTxn(txn)
            .logInfo()
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        String formXml = testParams['Test XML Data']
        Txn txn = new MockVoBuilder().createTxnTaskWithXml(formXml)
```

```
        Map params = [
            "svcDef": svcDef,
            "txn": txn
        ]

        new ServiceInvoker(svcDef).invoke(params)
    }
}
```

Tracking Number

The Tracking Number (tracking code) service is used to generate tracking codes for form transactions. The generated tracking code is included in the form XML data, and is also displayed on associated transaction pages and emails.

The generated Tracking Number value should contain alphanumeric values only and use upper case letters.

This service is configured via the Form Details tab under the Tracking Code section.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
form	the transaction record value object, see Form	no
request	the servlet HttpServletRequest	yes
user	the authenticated user, see User	yes

Script Result

The script should return a submissionDataBean which the submitted XML data.

Error Handling

If an error occurs invoking your Tracking Number Groovy script the error will be logged in the TM database error log, the user will be redirected to the error page on the form space. The transaction will be rolled back, and no transaction record will be created.

Service Template

This secion shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentTrackingNumberService {

    /*
     * Perform a transaction tracking number (tracking code) generation
     *
     * return: a globally unique tracking number value, values longer than 100 characters will be trimmed
     */
    String invoke(SvcDef svcDef, Form form, HttpServletRequest request, User user) {

        return new TrackingCodeBuilder()
            .setPrefix(form.formCode + "-")
            .setPostfix("-TEST")
            .build()
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.test.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        Form form = new MockVoBuilder().createForm()
```

```
        MockRequest request = new MockRequest()

        Map params = [
            "svcDef": svcDef,
            "form": form,
            "request": request,
            "user": null
        ]

        String trackingCode = (String) new ServiceInvoker(svcDef).invoke(params)

        logger.info "Tracking Code: " + trackingCode

        // Test any business logic performed
        assert trackingCode != null
        assert trackingCode.startsWith(form.formCode + "-")
        assert trackingCode.endsWith("-TEST")
    }
}
```

Transaction History Publisher

The Transaction History Publisher is used to publish transaction history data to customer's data warehouse.

The result list of ids should contain published records primary keys.

This service is configured via the Form Details tab under the Tracking Code section.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
rows	the rows data as List	no

Script Result

The script should return a list of numbers which are published.

Error Handling

If an error occurs invoking your Transaction History Publisher script the error will be logged in the TM database error log, the user will be redirected to the error page on the form space. The transaction will be rolled back, and no transaction record will be created.

Service Template

This section shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*
import com.jcraft.jsch.*

class FluentTransactionHistoryPublisher {

    /*
     * Perform a transaction history publishing
     *
     * return: list of the published transactions' primary keys.
     */
    List invoke(SvcDef svcDef, List<map> rows) {

        /* // JCraft SFTP Upload Example
        ChannelSftp channelSftp
        Session session
        try {
            String username = svcDef.svcConn.username
            String password = svcDef.svcConn.password
            String endpoint = svcDef.svcConn.endpoint

            JSch jsch = new JSch()
            session = jsch.getSession(username, endpoint, 22)
            session.setPassword(password)

            Properties config = new Properties()
            config.put("StrictHostKeyChecking", "no")
            session.setConfig(config)
            session.connect()
            Channel channel = session.openChannel("sftp")
            channel.connect()
            channelSftp = (ChannelSftp) channel

            // TODO: write the rows in local 'myfile.data' file and then upload
            channelSftp.put('/local/storage/myfile.data', '/remote/myfile.data')

        } finally {
            channelSftp?.disconnect()
            session?.disconnect()
        }
        */

        //Fill and return rows' ids
        final List result = new ArrayList<>()
        for (Map row: rows) {
            result.add(Long.parseLong(row.get('transaction_history_oid').toString()))
```

```
        }
        return result
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        //TODO fill some rows data
        List<map> rows = new ArrayList<>()

        Map params = [
            "svcDef": svcDef,
            "rows": rows
        ]

        List ids = (List) new ServiceInvoker(svcDef).invoke(params)

        logger.info "Published ids.size: " + ids.size()

        // Test any business logic performed
        assert ids != null
    }
}
```

Virus Scan

Virus Scan

The Virus Scan service is used to scan the provided file and return true if virus free.

The result is true if the file name and data are virus free, otherwise the result is false.

This service is configured via the Form Details tab under Virus Scan tab.

Service Invoke Parameters

Parameter	Description	Nullable
svcDef	the service definition value object, see SvcDef	no
fileName	the file name as String	no
fileData	the file data as byte[]	no

Script Result

The script should return boolean value - if true then the file is virus free, otherwise return false.

Runtime exception may occur if there is configuration error.

Error Handling

If an error occurs invoking your Virus Scan Groovy script the error will be logged in the TM database error log, the user will be redirected to the error page on the form space. The transaction will be rolled back, and no transaction record will be created.

Service Template

This secion shows service template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.vo.*

class FluentVirusScanService {

    /*
     * Perform a file virus scan
     *
     * return: true if the file is virus free
     */
    boolean invoke(SvcDef svcDef, String fileName, byte[] fileData) {

        return !fileName.endsWith('.exe')
    }
}
```

Unit Test Template

This section shows unit test template Groovy script.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

class UnitTest {

    /*
     * Perform service unit test
     *
     * throws exception if unit test fails
     */
    void invoke(SvcDef svcDef, Map testParams) throws Exception {

        Map params = [
            "svcDef": svcDef,
            "fileName": "filename.txt",
            "fileData": "some content".getBytes()
```

```
        ]

        boolean result = (boolean) new ServiceInvoker(svcDef).invoke(params)

        logger.info "Virus free: " + result

        assert result == true
    }
}
```

SSO Revalidation

This type of service is not yet supported in the Transact Fluent SDK. In the interim the legacy API can be used.

Provides a SSO revalidation script which if returns true will re-trigger the SSO to re authenticate the user. If the script returns true the SSO Authentication scripts are run again, if false the users current login session is used. When the revalidation script returns true, the Get SSO Auth script will execute and the authentication provider will run.

This script is enabled by selecting **Enable SSO Filter** and **Enable SSO Revalidation** check boxes on the Security Manager tab. This shows the SSO Revalidation Tab where the script can be modified.

Where the execution path returns true, the script should logout from the spring security context. This will invalidate the current session. You maybe required to copy session attribute from the existing session, logout then write them to the new session. This is how to logout from the spring security context.

```
SecurityContextLogoutHandler securityContextLogoutHandler = new SecurityContextLogoutHandler();
securityContextLogoutHandler.logout(request, null, null);
```

The intent of this script is to look at changes in the request headers which can trigger the re-authentication process. Examples:

- Checking the "referer" header, revalidate if this is not the coming from a TM Form Space or Federated Endpoint
- Checking if a header that holds the user login name against the currentUserAuthentication.getUsername()

This Groovy script is executed by the SSOAuthenticationFilter.

Script Interface

```
/* Provides a Groovy script to determine whether the requests session requires revalidation.

   Script parameters include:
       request : <a target="_blank" href="http://docs.oracle.com/javaee/7/api/javax/servlet/http
/HttpServletRequest.html">HttpServletRequest</a>
       currentUserAuthentication : <a target="_blank" href="http://static.springsource.org/spring-security/site
/docs/3.1.x/apidocs/org/springframework/security/core/Authentication.html">Authentication</a>
       portal : <a target="_blank" href="../../javadoc/com/avoka/fc/core/entity/Portal.html">Portal</a>
       securityManager : <a target="_blank" href="../../javadoc/com/avoka/fc/core/entity/SecurityManager.html"
>SecurityManager</a>

   Script return:
       true is revalidation is required, otherwise false. NOTE: script must return a boolean result.
*/
```

Service Invoke Parameters

Parameter	Description	Optional
request	the HTTP servlet request	no
currentUserAuthentication	the current users SpringSecurity authentication token	no
portal	the portal associated with the user's request	no
securityManager	the SecurityManager configuration entity	no

Error Handling

This script should generally not throw any errors. It should simply return true if re-authentication is required or false otherwise. Any errors thrown will be logged to the TM Error Log table by the **`SSOAuthenticationFilter`**.

Examples

The example script below will require re-authentication if the referer header has changed. This can be useful in the scenario where a user opening a new form on a clients web site, should be re-authenticated to ensure we have their latest profile information for form prefill.

```
/* Provides a Groovy script to determine whether the requests session requires revalidation.

   Script parameters include:
       request : javax.servlet.http.HttpServletRequest
       currentUserAuthentication : org.springframework.security.core.Authentication
       portal : com.avoka.fc.core.entity.Portal
       securityManager : com.avoka.fc.core.entity.SecurityManager
```

```
   Script return:
       true is revalidation is required, otherwise false
*/
import com.avoka.core.groovy.GroovyLogger as logger
import org.apache.commons.lang.StringUtils
import com.avoka.fc.core.service.EventLogService

EventLogService eventLogService = new EventLogService()

def logEvent = { msg ->
        if (false) {
          eventLogService.logInfoEvent("SSO Revalidation Script: " + msg, request)
        }
}

def msg = ""

String referer = request.getHeader("referer")

if (StringUtils.isBlank(referer) || StringUtils.isBlank(portal.getContextPath())){
  msg += "\n either referer or portal context path is blank. Revalidate=false"
  logEvent(msg)
  return false
}

if( referer.toLowerCase().startsWith(portal.getContextPath().toLowerCase())) {
  msg += "\n referer is from the portal. Revalidate=false"
  logEvent(msg)
  return false
}

if( referer.toLowerCase().startsWith("https://{adfs server domain name}/adfs")) {
  msg += "\n referer is from the federated endpoint. Revalidate=false"
  logEvent(msg)
  return false
}

msg += "\n referer is from a separate URL. Loging out spring security context. Revalidate=true"

logEvent(msg)

return true
```

SSO Get Authentication Token

This type of service is not yet supported in the Transact Fluent SDK. In the interim the legacy API can be used.

This service provides a script to retrieve an SSO token from a request. It is called by the TM SSOAuthenticationFilter to obtain the **SSOAuthenticationToken** when an unauthenticated request attempts to access a protected resource.

This script is configured via the Security Manager 'SSO Auth Filter' tab.

Script Interface

```
/** Provides a Groovy script to get an SSOAuthenticationToken from a request.
    The returned SSOAuthenticationToken will then be processed by the configured AuthenticationProvider(s).

    Script parameters include:
        request : <a target="_blank" href="http://docs.oracle.com/javaee/7/api/javax/servlet/http
/HttpServletRequest.html">HttpServletRequest</a>
        portal : <a target="_blank" href="../../javadoc/com/avoka/fc/core/entity/Portal.html">Portal</a>
        securityManager : <a target="_blank" href="../../javadoc/com/avoka/fc/core/entity/SecurityManager.html"
>SecurityManager</a>

    Script return:
        SSO auth token, or if null then other Authentication filters will be executed : <a target="_blank"
href="../../javadoc/com/avoka/fc/core/security/SSOAuthenticationToken.html">SSOAuthenticationToken</a>

    Script throws:
        redirect exception to redirect to an external login page : <a target="_blank" href="../../javadoc/com
/avoka/fc/core/servlet/RedirectException.html">RedirectException</a>
*/
```

Service Invoke Parameters

Parameter	Description	Optional
request	the HTTP request made by the user	no
portal	the portal associated with the user's request	no
securityManager	the security manager configuration entity	no

Error Handling

This Groovy script is executed by the SSOAuthenticationFilter. If the SSO authentication token is not present in the request, the script should redirect the user to the external authentication provider's login page using a RedirectException.

If there is a different system authentication error, the script can throw an AuthenticationException. The SSOAuthenticationFilter will then clear the SecurityContextHolder and set the exception in the request attribute WebAttributes.AUTHENTICATION_EXCEPTION.

Example

The example script below gets the HTTP session cookie **IM-AUTH-TOKEN** from the request. If this value is present then the user has been logged in by the SSO identity manager. If this cookie is not present, the user is redirected away to the SSO identity manager login page using a RedirectException.

In the case when the user is authenticated, the script gets the users login name and some user profile attributes from secure cookie values. Then it creates an SSOAuthenticationToken with the username and profile attributes set and returns the authentication token. This SSOAuthenticationToken is subsequently passed to the configured AbstractUserDetailsAuthenticationProvider which will perform user account creation steps if required and establish the login session.

Once the Spring login session has been established, this script will not need to be executed again until the user has logged out.

```
/** Provides a Groovy script to get an SSOAuthenticationToken from a request.
    The returned SSOAuthenticationToken will then be processed by the configured AuthenticationProvider(s).

    Script parameters include:
        request : javax.servlet.http.HttpServletRequest
        portal : com.avoka.fc.core.entity.Portal

    Script return:
        SSO auth token, or if null then other Authentication filters will be executed : com.avoka.fc.core.
security.SSOAuthenticationToken

    Script throws:
        redirect exception to redirect to an external login page : com.avoka.fc.core.servlet.RedirectException
```

```
*/
import org.apache.click.util.ClickUtils
import com.avoka.fc.core.security.SSOAuthenticationToken
import com.avoka.fc.core.servlet.RedirectException

def authToken = ClickUtils.getCookieValue(request, 'IM-AUTH-TOKEN')

if (authToken != null) {
    def username = ClickUtils.getCookieValue(request, 'IM-USER-ID')

    def attributes = [:]
    attributes['email'] = ClickUtils.getCookieValue(request, 'IM-USER-EMAIL')
    attributes['firstName'] = ClickUtils.getCookieValue(request, 'IM-USER-FIRST-NAME')
    attributes['lastName'] = ClickUtils.getCookieValue(request, 'IM-USER-LAST-NAME')

    return new SSOAuthenticationToken(username, attributes)

} else {
    throw new RedirectException('https://login.mycorp.com/?return=forms.mycorp.com/forms/secure/account/home.
htm')
}
```

SSO Authentication OK Response

This type of service is not yet supported in the Transact Fluent SDK. In the interim the legacy API can be used.

Provides a SSO authentication response script which can render the HTTP response if this is required to perform login flow customization.

If the script writes directly to the HTTP servlet response object, it should return false to indicate to the SSO Auth Filter that request processing has been completed.

This script is configured via the Security Manager 'SSO Auth Filter' tab.

Script Interface

```
/** Provides a Groovy script successful authentication event handler which can optionally
    write to the Servlet response and return false to signal that the request processing has been completed.

    Script parameters include:
        request : <a target="_blank" href="http://docs.oracle.com/javaee/7/api/javax/servlet/http
/HttpServletRequest.html">HttpServletRequest</a>
        response : <a target="_blank" href="http://docs.oracle.com/javaee/7/api/javax/servlet/http
/HttpServletResponse.html">HttpServletResponse</a>
        portal : <a target="_blank" href="../../javadoc/com/avoka/fc/core/entity/Portal.html">Portal</a>
        securityManager : <a target="_blank" href="../../javadoc/com/avoka/fc/core/entity/SecurityManager.html"
>SecurityManager</a>

    Script return:
        true to continue processing, or false to signal request processing has completed : boolean

    Script throws:
        redirect exception to redirect to an external page : <a target="_blank" href="../../javadoc/com/avoka/fc
/core/servlet/RedirectException.html">RedirectException</a>
*/

return true
```

Service Invoke Parameters

Parameter	Description	Optional
request	the HTTP servlet request	no
response	the HTTP servlet response	no
portal	the portal associated with the user's request	no
securityManager	the SecurityManager configuration entity	no

Error Handling

This Groovy script is executed by the SSOAuthenticationFilter. This script can redirect to another resource if required RedirectException.

If there is a different system authentication error the script can throw a AuthenticationException. The SSOAuthenticationFilter will then clear the SecurityContextHolder and set the exception in the request attribute WebAttributes.AUTHENTICATION_EXCEPTION.

SSO Authentication Provider

This type of service is not yet supported in the Transact Fluent SDK. In the interim the legacy API can be used.

Provides an Groovy script encapsulating SSO authentication logic which is executed by the GroovyUserDetailsAuthenticationProvider configured for the security manager.

This script is configured via the Security Manager's 'Authentication Providers' tab.

Script Interface

```
/** Provides a Groovy script to return the AccountUserDetails for the given loggin attempt.
    The returned SSOAuthenticationToken will then be processed by the configured AuthenticationProvider(s).

    Script parametes include:
        username : string
        authentication : <a target="_blank" href="../../javadoc/com/avoka/fc/core/security
/SSOAuthenticationToken.html">SSOAuthenticationToken</a>
        authParameters: <a target="_blank" href="http://docs.oracle.com/javase/7/docs/api/java/util/Map.html"
>Map</a><String, String>
        portal : <a target="_blank" href="../../javadoc/com/avoka/fc/core/entity/Portal.html">Portal</a>
        securityManager : <a target="_blank" href="../../javadoc/com/avoka/fc/core/entity/SecurityManager.html"
>SecurityManager</a>

    Script return:
        the user account details : <a target="_blank" href="../../javadoc/com/avoka/fc/core/security
/AccountUserDetails.html">AccountUserDetails</a>

    Script throws:
        <a target="_blank" href="http://static.springsource.org/spring-security/site/docs/3.1.x/apidocs/org
/springframework/security/authentication/BadCredentialsException.html">BadCredentialsException</a> : if the user
credentials were invalid
        <a target="_blank" href="http://static.springsource.org/spring-security/site/docs/3.1.x/apidocs/org
/springframework/security/core/userdetails/UsernameNotFoundException.html">UsernameNotFoundException</a> : if
the user was not found
        <a target="_blank" href="http://static.springsource.org/spring-security/site/docs/3.1.x/apidocs/org
/springframework/security/authentication/AuthenticationServiceException.html">AuthenticationServiceException</a>
: if a system authentication service error occured
        <a target="_blank" href="../../javadoc/com/avoka/fc/core/security/NotPortalAccountException.html"
>AccountNotActiveException</a> : if the user account is not active
        <a target="_blank" href="../../javadoc/com/avoka/fc/core/security/NotPortalAccountException.html"
>NotPortalAccountException</a> : if the user account is not associated with the portal
*/
```

Service Invoke Parameters

Parameter	Description	Optional
username	the user login name, or login identifier	no
authentication	the SSO authentication token	no
authParameters	a map of Authentication Provider configuration parameter values for the , keyed on parameter name	no
portal	the portal associated with the user's request	no
securityManager	the SecurityManager configuration entity	no

Error Handling

If an unexpected system error occurs your script should throw a AuthenticationServiceException which will be recorded in the TM database error log.

Other exceptions are used convey to authentication attempt failure information:

- BadCredentialsException : if the user credentials werer invalid
- UsernameNotFoundException : if the user was not found
- AccountNotActiveException : if the user account is not active
- NotPortalAccountException : if the user account is not associated with the portal

Example

The script below provides an example SSO authentication provider script. Please note this script assumes the user has been successfully authenticated by a separate SSO identity management system, and a valid SSO authentication token is provided to this service.

This script performs a lookup to see if the linking SSO user account already exists in the TM database.

If the user account is found, a Spring AccountUserDetails object referencing the user account record and the granted authorities (groups) from the authentication token is created. This object will then be used to initialize the authenticated user session. The granted authorities can be used to enable form group access control to restricted forms by mapping provided SSO groups onto TM form groups.

If a linking SSO user account doesn't exist in the TM database, one is created using the UserService.createSsoUserAccount method. The returned user account object is used to initialize an **`AccountUserDetails`** object which is then returned.

```
/** Provides a Groovy script to return the AccountUserDetails for the given log-in attempt.
    The returned SSOAuthenticationToken will then be processed by the configured AuthenticationProvider(s).

    Script parametes include:
        username : string
        authentication : com.avoka.fc.core.security.SSOAuthenticationToken
        portal : com.avoka.fc.core.entity.Portal

    Script return:
        the user account details : com.avoka.fc.core.security.AccountUserDetails

    Script throws:
        org.springframework.security.authentication.BadCredentialsException : if the user credentials were
invalid
        org.springframework.security.core.userdetails.UsernameNotFoundException : if the user was not found
        org.springframework.security.authentication.AuthenticationServiceException : if a system authentication
service error occured
        com.avoka.fc.core.security.AccountNotActiveException : if the user account is not active
        com.avoka.fc.core.security.NotPortalAccountException : if the user account is not associated with the
portal
*/
import com.avoka.fc.core.dao.UserAccountDao
import com.avoka.fc.core.service.ServiceFactory
import com.avoka.fc.core.security.AccountUserDetails
import com.avoka.fc.core.security.AccountNotActiveException
import org.springframework.security.authentication.AuthenticationServiceException

// Exit early if no authentication token present
if (authentication == null) {
    throw new AuthenticationServiceException("Missing authentication token")
}

// Get get user profile information from authentication token attributes
def attributes = authentication.getAttributes()

def email = attributes["email"]
def firstName = attributes["firstName"]
def lastName = attributes["lastName"]

def profileMap = [:]
profileMap["Email"] = email
profileMap["Given Name"] = firstName
profileMap["Family Name"] = lastName

// Get the users granted authorities (TM Groups) from authentication token
def authorities = authentication.getAuthorities()

def userService = ServiceFactory.getUserService(portal)

def userAccountDao = new UserAccountDao()
def userAccount = userAccountDao.getActiveUserAccountForLogin(username)

// Found user ensure not locked and update profile and portal association
if (userAccount != null) {
    // ensure a temporary lock is cleared if needed
    userService.updateLockStatus(userAccount)

    if (!userAccount.isActive()) {
        throw new AccountNotActiveException("Account not active: ", userAccount.getAccountStatus())
    }

    if (userAccount.isEmailVerificationRequired()) {
        throw new AccountNotActiveException("Account requires email verification", "")
    }

    userService.updateActiveUserProfile(userAccount, profileMap)
```

```
    userService.addPortalForUser(userAccount, portal)

    return new AccountUserDetails(userAccount, authorities)
}

// User account not found, create account
def newAccount = userService.createSsoUserAccount(username, email, firstName, lastName, profileMap)

return new AccountUserDetails(newAccount, authorities)
```

Transact REST API Reference

Avoka Transact provides a set of REST API's can be used by external systems to perform integration by calling Transaction Manager.

These REST APIs include:

- Application Package API which provides a REST application package API
- Delivery API which provides a REST transaction delivery API
- Form Groups API which provides a REST Form Groups API
- Groovy Service Invoke which provides a REST Groovy Service invocation endpoint
- Service Definitions API which provides a REST Service Definitions API
- Tasks API which provides a REST form Tasks API
- TPac API which provides a REST TPac API
- Transactions API which provides a REST form Transactions API
- Transaction History API which provides a REST form Transaction History API

Application Package API

REST Application Package API

The Application Package API provides a REST service for listing application packages in TM and uploading and downloading application package archive files.

Security

To access the Application Package REST endpoint the caller needs to be authenticated using HTTP Basic Authentication. The authenticated user will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission '`REST Application Package API`'.

If the user is not a global administrator, only application packages belonging to the organizations assigned to the user will be accessible.

URL Endpoint

The REST application package URL endpoint is as follows:

```
https://<TM server>/manager/secure/rest/application-package/v1/
```

Application Package API

This section provides a description all the REST Application Package API operations including:

- GET Application Package List
- GET Application Package Archive
- POST Upload Application Package Archive (New Application)
- PUT Upload Application Package Archive (Existing Application)
- DELETE Application Package

GET Application Package List

Return the list of application packages for an organization, sorted by name.

URL Format

```
GET https://<tm server>/manager/secure/rest/application-package/v1/<client code>
```

The URL needs to include the client code of the organization for which application packages are to be listed.

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/application-package/v1/maguire
```

200 Response

```
[
  {
    "id": 8,
    "applicationName": "My Application 1",
    "createdAt": "2013-12-23T00:00+1100",
    "createdBy": "admin1",
    "description": "This is the main application.",
    "clientCode": "maguire",
    "clientName": "Maguire",
    "forms": [
      {
        "clientFormCode": "basic-form",
        "description": "This is a basic form that can be hosted by SFM. It is available as HTML and PDF and also
includes a PDF receipt.",
        "name": "Basic Form - PDF"
      },
      {
        "clientFormCode": "pf-maguire",
        "name": "Payment Form"
      },
```

```
    {
      "clientFormCode": "spdf-maguire",
      "name": "Static PDF Form"
    }
  ],
  "organizationProperties": [
    {
      "dataType": "Image",
      "propertyName": "Logo",
      "value": "4.jpg"
    },
    {
      "dataType": "String",
      "propertyName": "URL",
      "value": ""
    },
    {
      "dataType": "String",
      "propertyName": "Client Property maguire",
      "value": "This is a test property value."
    },
    {
      "dataType": "HTML",
      "propertyName": "Additional Services",
      "value": ""
    }
  ],
  "deliveryChannels": [
    {
      "deliveryMethod": "Delivery Process",
      "description": "",
      "name": "Shredder"
    },
    {
      "deliveryMethod": "Delivery Process",
      "description": "",
      "name": "Client Process Delivery"
    }
  ],
  "serviceConnections": [
    {
      "clientCode": "maguire",
      "name": "JBoss Maguire",
      "serverType": "JBoss"
    }
  ],
  "serviceDefinitions": [
    {
      "description": "Provides form lookup form data service by calling a configurable Groovy script",
      "name": "Client Service 2",
      "type": "Dynamic Data",
      "versionNumber": 1
    },
    {
      "description": "Provides a Groovy script based submission delivery process.",
      "name": "Export Groovy Delivery Process",
      "type": "Delivery Process",
      "versionNumber": 1
    },
    {
      "description": "",
      "name": "Client Service 1",
      "type": "Receipt Number",
      "versionNumber": 3
    },
    {
      "description": "",
      "name": "Client Service 1",
      "type": "Receipt Number",
      "versionNumber": 4
    }
  ]
},
{
  "id": 2,
  "applicationName": "My Application 2",
  "createdAt": "2013-12-23T00:00+1100",
  "createdBy": "admin2",
```

```
    "description": "",
    "clientCode": "maguire",
    "clientName": "Maguire",
    "forms": [
      {
        "clientFormCode": "pf-maguire",
        "name": "Payment Form"
      }
    ],
    "deliveryChannels": [],
    "serviceDefinitions": [
      {
        "description": "Provides form lookup form data service by calling a configurable Groovy script",
        "name": "Client Service 4",
        "type": "Dynamic Data",
        "versionNumber": 1
      },
      {
        "description": "",
        "name": "Client Service 1",
        "type": "Receipt Number",
        "versionNumber": 1
      },
      {
        "description": "",
        "name": "Client Service 1",
        "type": "Receipt Number",
        "versionNumber": 4
      },
      {
        "description": "Demonstration payment gateway card payment service.",
        "name": "Private Demo Card Payment",
        "type": "Payment Gateway",
        "versionNumber": 1
      }
    ]
  }
]
```


GET Application Package Archive

Return the application package archive for the specified organization code and application package name.

URL Format

```
https://<tm server>/manager/secure/rest/application-package/v1/<client code>/<application name>/
```

Request Parameters

The table below shows the set of request parameters.

Parameter	Description
includeAllVersions	a boolean parameter that determines whether all form versions are included in the application archive, or only the current form version for each form (Values: true, false)

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/application-package/v1/maguire/my-app/
```

200 Response

The service will return an application package archive file. This is the same kind of archive file that can be obtained by exporting the application package from the TM management console.

Relevant response headers include:

```
Content-Disposition attachment;filename="application-package-archive-My_Application_1-2017-03-01.zip"
Content-Length 1573905
Content-Type application/zip
```

POST Upload Application Package Archive (New Application)

Upload an application package archive ZIP file into the TM server. The application package with the given name and organization must not exist on the server for the call to succeed.

URL Format

```
POST https://<tm server>/manager/secure/rest/application-package/v1/<client code>/
```

POST Body

The Multipart POST request parameters include:

- `archiveFile` - the application package archive ZIP file (required)
- `importDeliveryChannels` - boolean parameter that determines whether delivery channels will be included in the import. Values: true, false (default true)
- `importForms` - boolean parameter that determines whether forms will be included in the import. Values: true, false (default true)
- `importOrganizationProperties` - boolean parameter that determines whether organization properties will be included in the import. Values: true, false (default true)
- `importServiceConnections` - boolean parameter that determines whether service connections will be included in the import. Values: true, false (default true)
- `importServices` - boolean parameter that determines whether service definitions will be included in the import. Values: true, false (default true)
- `preserveDeliveryChannels` - boolean parameter that determines whether existing delivery channels will be preserved (not modified during import). Values: true, false (default false)
- `preserveServiceConnections` - boolean parameter that determines whether existing service connections will be preserved (not modified during import). Values: true, false (default false)
- `preserveServices` - boolean parameter that determines whether existing service definitions will be preserved (not modified during import). Values: true, false (default false)

Example

Please note POST example below is idealized to provide an illustration of multipart POST request.

```
POST /manager/secure/rest/application-package/v1/maguire/ HTTP/1.1
Host: https://transact.maguire.com
Content-type: multipart/form-data, boundary=AaB03x

--AaB03x
content-disposition: form-data; name="archiveFile"; filename="application-package-archive-Maguire_Services-2017-
03-01.zip"
Content-Transfer-Encoding: binary
...
--AaB03x--
=========================
```

200 Response

```
{
  "archiveName": "application-package-archive-Maguire_Services-2017-03-01.zip",
  "importMessage": "Application package named 'Maguire Services' for organization 'maguire' was imported
successfully.",
  "importStatus": "Completed",
  "importTime": "2017-03-01T13:38+1100"
}
```

PUT Upload Application Package Archive (Existing Application)

Upload an application package archive ZIP file into the TM server. The application package with the given name and organization must already exist on the server for the call to succeed.

URL Format

```
PUT https://<tm server>/manager/secure/rest/application-package/v1/<client code>/
```

PUT Body

The Multipart PUT request parameters include:

- `archiveFile` - the application package archive ZIP file (required)
- `importDeliveryChannels` - boolean parameter that determines whether delivery channels will be included in the import. Values: true, false (default true)
- `importForms` - boolean parameter that determines whether forms will be included in the import. Values: true, false (default true)
- `importServiceConnections` - boolean parameter that determines whether service connections will be included in the import. Values: true, false (default true)
- `importServices` - boolean parameter that determines whether service definitions will be included in the import. Values: true, false (default true)
- `preserveDeliveryChannels` - boolean parameter that determines whether existing delivery channels will be preserved (not modified during import). Values: true, false (default false)
- `preserveServiceConnections` - boolean parameter that determines whether existing service connections will be preserved (not modified during import). Values: true, false (default false)
- `preserveServices` - boolean parameter that determines whether existing service definitions will be preserved (not modified during import). Values: true, false (default false)

Example

Please note PUT example below is idealized to provide an illustration of multipart PUT request.

```
PUT /manager/secure/rest/application-package/v1/maguire/ HTTP/1.1
Host: https://transact.maguire.com
Content-type: multipart/form-data, boundary=AaB03x

--AaB03x
content-disposition: form-data; name="archiveFile"; filename="application-package-archive-Maguire_Services-2017-
03-01.zip"
Content-Transfer-Encoding: binary
...
--AaB03x--
==========================
```

200 Response

```
{
  "archiveName": "application-package-archive-Maguire_Services-2017-03-01.zip",
  "importMessage": "Application package named 'Maguire Services' for organization 'maguire' was imported
successfully.",
  "importStatus": "Completed",
  "importTime": "2017-03-01T13:39+1100"
}
```

DELETE Application Package

Irrevocably delete an application package. **USE THIS OPERATION WITH CAUTION**.

URL Format

```
https://<tm server>/manager/secure/rest/application-package/v1/<client code>/<application package name>
```

DELETE Examples

```
DELETE https://transact.maguire.com/manager/secure/rest/application-package/v1/maguire/my-app
```

200 Response

The response will not contain any additional information. If the status code is 200, the application package was deleted successfully.

Application Package JSON

The GET application package list operation described above returns application package value objects.
See the GET section for an example.

Application Package

Attribute	Type	Description
applicationName	String	the name of the application package
clientCode	String	the client code of the organization owning the application package
clientName	String	the name of the organization owning the application package
createdAt	Date	the application package creation date
createdBy	String	the login name of the user who created the application package
deliveryChannels	List of delivery channel value objects	the list of delivery channels associated with the application package
description	String	a description of the application package
forms	List of form value objects	the list of forms associated with the application package
id	Long	the OID of the application package
lastModifiedAt	Date	the last modified date of the application package
lastModifiedBy	String	the login name of the user who last modified the application package
organizationProperties	List of organization property value objects	the list of organization properties associated with the application package
serviceConnections	List of service connection value objects	the list of service connections associated with the application package
serviceDefinitions	List of service definition value objects	the list of service definitions associated with the application package

Delivery Channel

Attribute	Type	Description
deliveryMethod	String	the delivery method (e.g. Delivery Process)
description	String	the description for the delivery channel
name	String	the name of the delivery channel (unique within the organization)

Form

Attribute	Type	Description
clientFormCode	String	the unique form code for this form
description	String	the form description
name	String	the form name (unique within the organization)

Organization Property

Attribute	Type	Description
dataType	String	the data type of the property (e.g. String, HTML, Image)
propertyName	String	the name of the property (e.g. URL)
value	String	the property value (e.g. http://www.example.com).

Service Connection

Attribute	Type	Description
clientCode	String	the client code of the organization that the service connection belongs to (if any)
name	String	the name of the service connection
serverType	String	the server type (e.g. AWS S3)

Service Definition

Attribute	Type	Description
description	String	the description for the service
name	String	the name of the service definition

type	String	the service type (e.g. Dynamic Data)
versionNumber	Integer	the version number of the service definition

Delivery API

REST Delivery API

The Delivery Service provides a REST service for performing transaction data delivery integration with other systems. Using this service you can retrieve rich set of transaction data from remote Transaction Manager servers in a secure and efficient manner.

Delivery transaction data includes:

- form XML
- PDF receipt document
- form transaction metadata
- uploaded file attachments (optional)
- Excel XML data extract (optional)

Note: It is generally recommended that the REST delivery service should be in preference to the earlier SOAP Web Service delivery service, as it enables faster integration development, provides a richer data model and is more efficient.

Getting Started

This section discusses how to get started using the Delivery Service API.

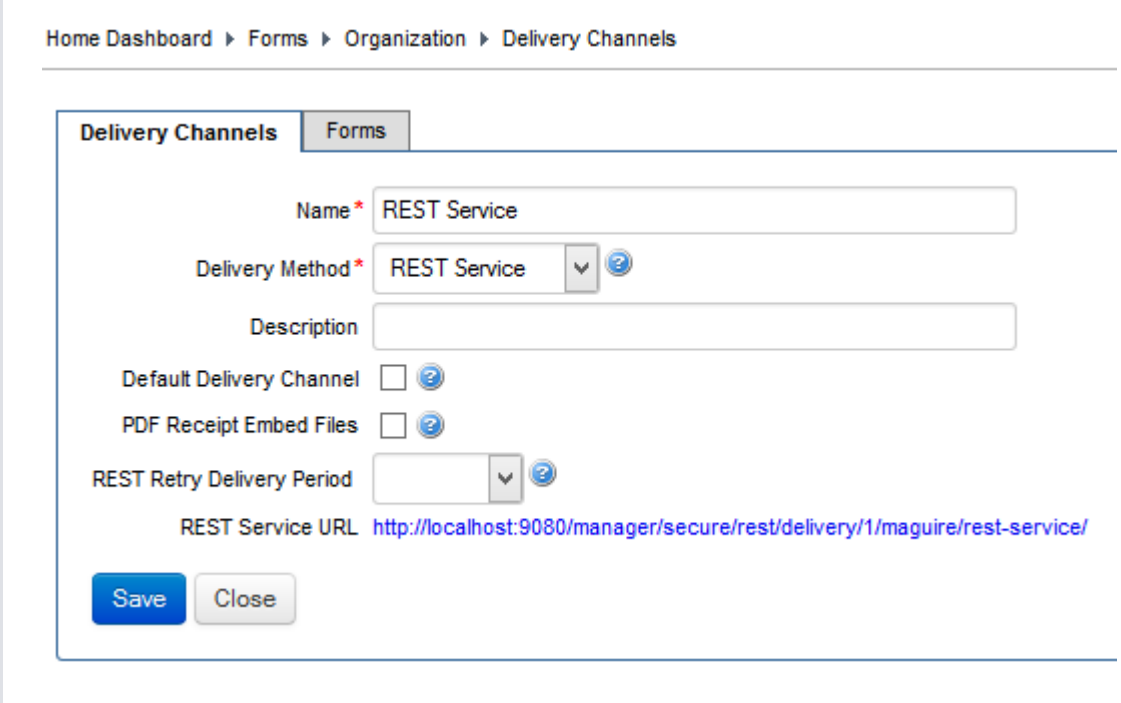
Security

To access the Delivery REST endpoint the caller needs to be authenticated using HTTP Basic Authentication. The authenticated user will need an user account on the TM server and this account will need to be Active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission '**`Rest Delivery Service API`**'. A standard TM role '**`REST Delivery`**' is provided which includes this permission, simply assign the user account this role.

Delivery Channel

To make form transactions available for delivery via the REST Delivery Service you need to create a REST Service delivery channel for the form's organization and associate your form with this delivery channel.



An organization can have multiple REST Service delivery channels defined one for each form, or they can have a single REST Service delivery channel which is configured as the default channel for the Organization. If a delivery channel is configured as the default channel, then any forms which don't have a delivery channel configured will use the default channel.

You should think of REST Service delivery channels as separate queues from which you can get completed transaction from.

URL Endpoint

The REST delivery service URL endpoint is provided as a link in the Delivery Channel configuration screen. The general format of this URL is provided below:

```
https://<tm server>/manager/rest/delivery/v1/<client code>/<channel name>/
```

Test Call

Once you have completed these steps you can call the Delivery Service URL directly using your browser, for example:

```
https://forms.mycorp.com/manager/secure/rest/delivery/v1/maguire/rest-service/
```

Once you have answered the basic authentication challenge you should see a JSON response like the one below.

```
{
  "Ready": [],
  "In Progress": [],
  "Error": []
}
```

This response is telling you that there are no submissions ready for delivery, are in-progress or in error.

Performing Delivery

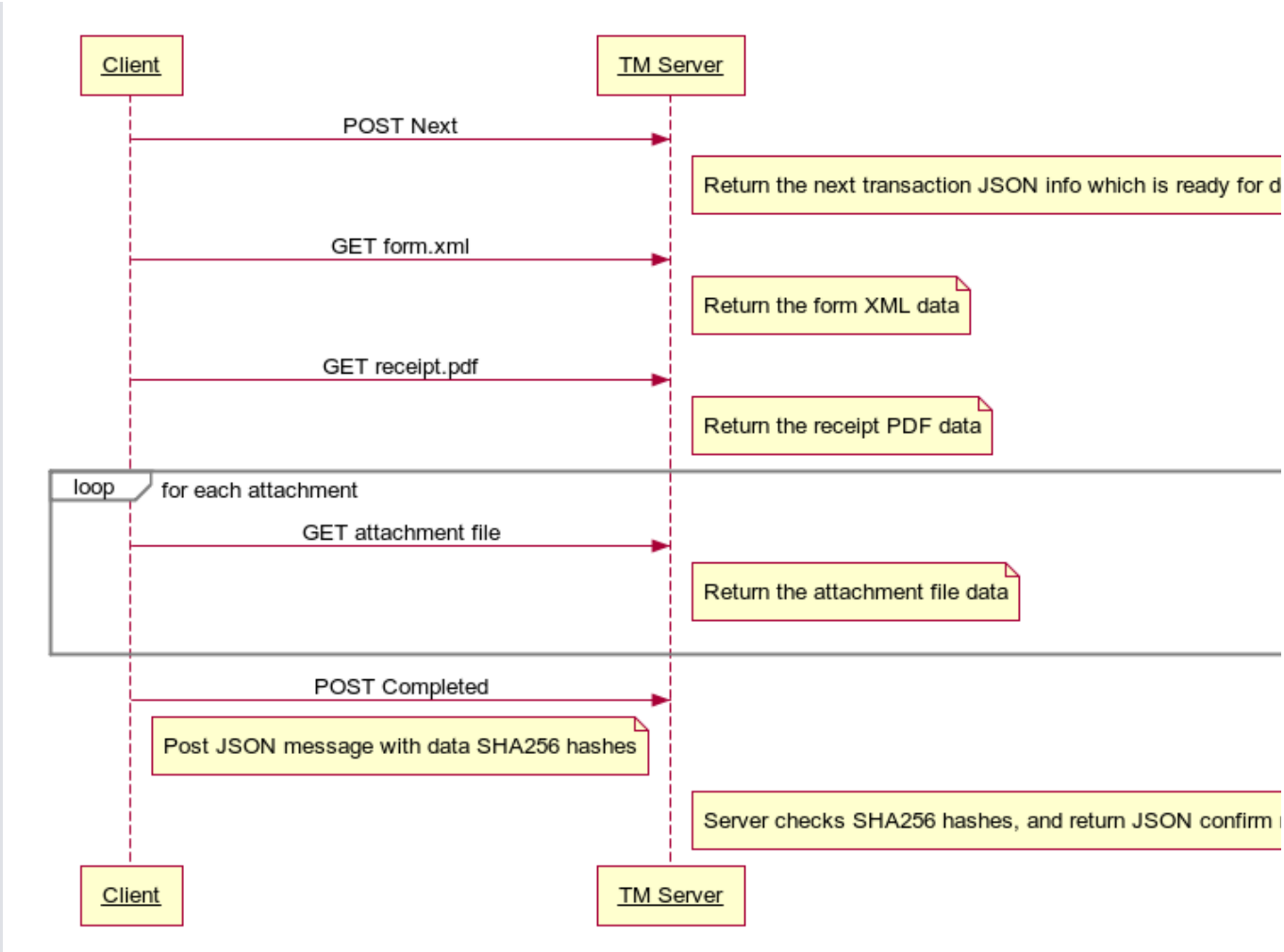
The REST Delivery service provide queue based delivery integration model which supports High Availability (HA) integration without having to do a lot of state management work with the integration clients. The basic calling pattern is:

1. get transaction item from the delivery queue
2. download transaction data: XML, PDF, file attachments, etc.
3. confirm transaction delivery

By using a queue based model, you can have multiple HA integration clients running, pulling down transactions from the same delivery queue. If one of the client fails, then other clients can continue to process transactions.

Call Sequence

The REST delivery call sequence diagram is illustrated below.



POST Next

The first REST operation to perform is to get the next transaction ready for delivery by making a POST request to the example URL below.

```
POST http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/next/
```

This operation will return a transaction from the delivery channel queue and change its Delivery Status to 'In Progress'. This call will return a JSON response containing the transaction meta data. An example of is data is provided below:


```
{
    "id": 30,
    "trackingCode": "TVKC6R",
    "receiptNumber": "FTX-CCA-1",
    "submitKey": "c0e695fac9941801fe9cb5572150d21f",
    "formCode": "FTX-CCA",
    "formName": "Credit Card Application",
    "formVersion": "1.0",
    "clientCode": "maguire",
    "clientName": "Maguire",
    "portal": "Maguire",
    "fieldWorkerFlag": false,
    "testMode": false,
    "contactEmailAddress": "jeff.johnson@avoka.com",
    "formStatus": "Completed",
    "timeRequest": "2015-02-06T15:41:25+11:00",
    "timeSubmission": "2015-02-06T15:41:46+11:00",
    "timeLastUserActivity": "2015-02-06T15:42:20+11:00",
    "timeFormCompleted": "2015-02-06T15:41:46+11:00",
    "ipAddress": "127.0.0.1",
    "requestCookie": "JSESSIONID=SpOSp+hJFvw8cdI30CJj-S6j; A983-2928-2398-3419=",
    "referer": "http://www.google.com",
    "userAgent": "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:35.0) Gecko/20100101 Firefox/35.0",
    "userAgentBrowser": "Firefox 35",
    "userAgentBrowserType": "Firefox",
    "userAgentBrowserVersion": "35",
    "userAgentDeviceType": "Desktop",
    "userAgentOs": "Windows 8.1",
    "userAgentOsType": "Windows",
    "userAgentOsVersion": "8.1",
    "dataValidationStatus": "Error",
    "dataValidationMessage": "Payment calculation error. Submitted payment amount $1.00, while calculated
payment amount $1260.59.",
    "dataExtracts": {
        "First Name": "Jeff",
        "Last Name": "Johnson",
        "Email": "jeff.johnson@avoka.com"
    },
    "deliveryStatus": "In Progress",
    "deliveryMethod": "REST Service",
    "deliveryMessage": "Delivered via 'REST Service' REST Service to rest-delivery from remote address
127.0.0.1",
    "deliveryChannel": "REST Service",
    "deliveryTime": "2015-02-06T15:43:32+11:00",
    "dataDeleted": false,
    "xmlURL": "http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/form.xml",
    "receiptURL": "http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/receipt.pdf",
    "extractURL": "http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/extract.xls",
    "attachments": [
        {
            "id": 15,
            "attachmentName": "Pay Slip",
            "requiredFlag": false,
            "fileName": "Greenbook_Final.pdf",
            "fileSize": 1852554,
            "fileURL": "http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/15
/Greenbook_Final.pdf",
            "contentType": "application/pdf",
            "timeUploaded": "2015-02-06T17:12:34+11:00"
        },
        {
            "id": 16,
            "attachmentName": "Employment Contract",
            "requiredFlag": false,
            "fileName": "AQR001113Solution Design.docx",
            "fileSize": 3297658,
            "fileURL": "http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/16
/AQR001113Solution Design.docx",
```

```
            "contentType": "application/vnd.openxmlformats-officedocument.wordprocessingml.template",
            "timeUploaded": "2015-02-06T17:12:38+11:00"
        }
    ]
}
```

Once a transaction has been take from the queue using the POST Next operation it will no longer be available on the queue.

GET Data

The using the transaction metadata the client application should now download the associate transaction data using the URL's specified in the transaction metadata. Examples of these transaction data URLs illustrated below.

```
{
  "id": 30,
  ...
  "xmlURL": "http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/form.xml",
  "receiptURL": "http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/receipt.pdf",
  "extractURL": "http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/extract.xls",
  "attachments": [
    {
      "id": 15,
      "attachmentName": "Pay Slip",
      "requiredFlag": false,
      "fileName": "Greenbook_Final.pdf",
      "fileSize": 1852554,
      "fileURL": "http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/15
/Greenbook_Final.pdf",
      "contentType": "application/pdf",
      "timeUploaded": "2015-02-06T17:12:34+11:00"
    },
    {
      "id": 16,
      "attachmentName": "Employment Contract",
      "requiredFlag": false,
      "fileName": "AQR001113Solution Design.docx",
      "fileSize": 3297658,
      "fileURL": "http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/16
/AQR001113Solution-Design.docx",
      "contentType": "application/vnd.openxmlformats-officedocument.wordprocessingml.template",
      "timeUploaded": "2015-02-06T17:12:38+11:00"
    }
  ]}
```

From this transaction metadata we can see that we need to make the following transaction data download requests:

```
GET http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/form.xml
GET http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/receipt.pdf
GET http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/extract.xls
GET http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/15/Greenbook_Final.pdf
GET http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/16/AQR001113Solution-Design.
docx
```

Clients which support HTTP compression should set the HTTP header "Accept-Encoding: gzip". The TM REST service will gzip compress all responses which are suitable for compression to improve network performance. Content which will not be compressed includes PDF, image, video and audio content.

POST Confirmation

Once all the transaction data has been downloaded from the server the client should POST a delivery confirmation JSON message to the server confirming the transaction data has been successfully delivered. The confirmation URL is derived from the delivery channel endpoint URL with the addition transaction submission id, which is derived from the transaction metadata.

```
{
  "id": <b>30</b>,
  ...
}
```

```
POST http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/<b>30</b>/
```

The JSON confirmation message posted to the server must contain SHA-256 hex encoded hash values of the form XML data and any file attachments. The attachments IDs are specified in the transaction metadata attachments array. These attachments IDs correspond to the TM server Attachment IDs.

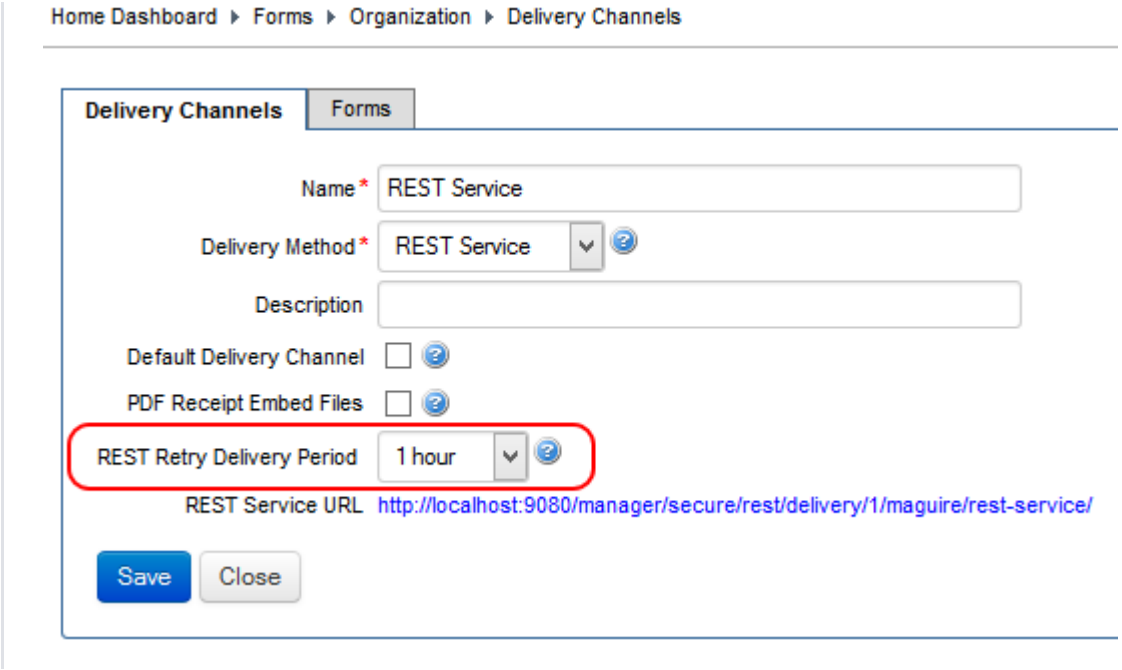
```
{
    "xmlSHA256": "bd2f85644beffdd101be4137c68269627bbb935d8cd231ab2e5c505eabff72c6",
    "attachments": [
        {
            "id": 15,
            "fileSHA256": "3b80d18bc78aad01c50efbad9355a48a6907bdb850b96fc919570209e8cfc4c4"
        },
        {
            "id": 16,
            "fileSHA256": "553a417def543b52bcc786f1e2b79dbfd958348c94f689e8d85181aacd039a61"
        }
    ],
    "deleteData": true,
    "processingStatus": "Your application is being processed"
}
```

To calculate the SHA-256 hex encoded values there are a variety of open source libraries available. Below is a simple Java function which will create a SHA-256 data hash and Hex encodes the value.

```
// Hexidecimal characters for encoding.
private static final char[] HEXADECIMAL = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c',
'd', 'e', 'f' };

public static String toSHA256Hash(byte[] bytes) {
    if (bytes == null) {
        throw new IllegalArgumentException("Null bytes parameter");
    }

    try {
        MessageDigest md = MessageDigest.getInstance("SHA-256");

        md.update(bytes);

        byte[] binaryData = md.digest();

        // Hex encode data SHA-256 digest
        char[] buffer = new char[64];

        for (int i = 0; i < 32; i++) {
            int low = (binaryData[i] & 0x0f);
            int high = ((binaryData[i] & 0xf0) >> 4);
            buffer[i * 2] = HEXADECIMAL[high];
            buffer[i * 2 + 1] = HEXADECIMAL[low];
        }

        return new String(buffer);

    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
```

Client Failure Recovery

If a client fails while processing a delivery (e.g. it has taken a transaction off the queue, which is now in 'In Progress' delivery state but hasn't completed the delivery), then this transaction is not available for other clients to pickup from the queue again. To enable retrying a delivery the REST Service Delivery Channel enables configuring a delivery retry timeout, after which any 'In Progress' transactions are automatically returned to a 'Ready' state so they are available again on the delivery queue.



Service API

This section provides a description all the REST Delivery Service API operations including:

- GET Clients
- GET Delivery Channels
- GET Submissions
- GET Submission
- GET Submission Data
- POST Next Submission
- POST Delivery Completed
- POST Delivery Error

GET Clients

Return the list of client codes available for the current users. Note global users would see a listing of all clients.

URL Format

```
GET https://<tm server>/manager/secure/rest/delivery/1/
```

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/delivery/1/
```

200 Response

```
[
    "maguire",
    "first-bank",
    "state-bank"
]
```

GET Delivery Channels

Return the list of delivery channel names for the specified client.

URL Format

```
GET https://<tm server>/manager/secure/rest/delivery/1/<client code>
```

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/delivery/v1/maguire/
```

200 Response

```
[
    "credit-card",
    "home-loans",
    "home-loans-abandoned",
    "insurance",
    "insurance-abandoned"
]
```

GET Submissions

Return the list of submissions id, for the specified client and delivery channel, with the status delivery "Ready", "In Progress" and "Error".

URL Format

```
GET https://<tm server>/manager/secure/rest/delivery/1/<client code>/<delivery channel>/
```

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/delivery/v1/maguire/credit-card/
```

200 Response

```
{
    "ready": [
        "2301",
        "2302",
        "2303",
        "2304",
        "2305"
    ],
    "inProgress": [
        "2300"
    ],
    "error": []
}
```

Query Parameters

This operation will also support a deliveryStatus query parameters, which can be used to filter on [Ready | InProgress | Error] values. Query parameter values will not be case sensitive. For example the following request:

```
GET https://transact.maguire.com/manager/secure/rest/delivery/v1/maguire/credit-card/?deliveryStatus=Ready
```

Will return the JSON response:

```
[
    "2301",
    "2302",
    "2303",
```

```
    "2304",
    "2305"
]
```

GET Submission

Return the submission for the specified submission id, client and delivery channel.

URL Format

```
GET https://<tm server>/manager/secure/rest/delivery/1/<client code>/<delivery channel>/<submission id>/
```

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/delivery/v1/maguire/credit-card/2039/
```

200 Response

The comprehensive submission example JSON response is provided below.


```
{
  "id": 69,
  "trackingCode": "UD65H6",
  "receiptNumber": "FTX-CCA-16",
  "submitKey": "20bc0dd18c74ae4f4e714dc402728d96",
  "taskKey": "a7d7c46f590c87903e39f3eb7cc2cb06",
  "taskType": "Review",
  "taskSubject": "Review Job Example 1 by medgar@avoka.com.",
  "taskMessage": "Please review the Job Example 1 by Malcolm Edgar.",
  "taskCreatedTimestamp": "2015-01-30T15:22:20+11:00",
  "job": {
    "id": 1,
    "jobName": "1 Step Review Job",
    "referenceNumber": "ZMYS4A",
    "jobKey": "24e67afe8ad22a92b1ebc3aa3dcad90c",
    "jobStep": "Application Review",
    "jobAction": "Assign Review"
  },
  "formCode": "FTX-CCA",
  "formName": "Credit Card Application",
  "formVersion": "1.0",
  "clientCode": "maguire",
  "clientName": "Maguire",
  "portal": "Maguire",
  "fieldWorkerFlag": false,
  "testMode": false,
  "contactEmailAddress": "jeff.johnson@avoka.com",
  "formStatus": "Completed",
  "timeRequest": "2015-01-27T14:51:38+11:00",
  "timeSubmission": "2015-01-27T14:52:04+11:00",
  "timeLastUserActivity": "2015-01-27T14:52:14+11:00",
  "timeFormCompleted": "2015-01-27T14:52:04+11:00",
  "ipAddress": "127.0.0.1",
  "requestCookie": "JSESSIONID\u003dPCgDnYPkLB+H6EHsXt2t6RHe; A983-2928-2398-3419\u003d",
  "referer": "http://localhost:9080/maguire/servlet/SmartForm.html?formCode\u003dFTX-CCA",
  "userAgent": "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:35.0) Gecko/20100101 Firefox/35.0",
  "userAgentBrowser": "Firefox 35",
  "userAgentBrowserType": "Firefox",
  "userAgentBrowserVersion": "35",
  "userAgentDeviceType": "Desktop",
  "userAgentOs": "Windows 8.1",
  "userAgentOsType": "Windows",
  "userAgentOsVersion": "8.1",
  "paymentStatus": "Completed",
  "paymentType": "Credit/Debit Card",
  "paymentTotal": 23.5,
  "paymentLog": {
    "paymentStatus": "Completed",
```

```
    "paymentLogKey": "9cb2345c50968900ac7d05e3751e465f",
    "paymentServiceCode": "DEMO",
    "userIpAddress": "127.0.0.1",
    "doTimestamp": "2015-01-27T15:10:49+11:00",
    "doAmount": 2350,
    "doMerchTxnRef": "1",
    "doVersion": "1",
    "drTimestamp": "2015-01-27T15:10:49+11:00",
    "drAmount": 2350,
    "drMerchTxnRef": "1",
    "drReceiptNo": "27",
    "drTxnResponseCode": "0",
    "drTxnResponseMsg": "Transaction approved"
  },
  "attachmentsStatus": "Completed",
  "dataExtracts": {
    "First Name": "Jeff",
    "Last Name": "Johnson",
    "Email": "jeff.johnson@avoka.com"
  },
  "deliveryStatus": "Ready",
  "dataDeleted": false,
  "xmlURL": "http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/69/form.xml",
  "receiptURL": "http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/69/receipt.pdf",
  "extractURL": "http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/69/extract.xls",
  "attachments": [
    {
      "id": 41,
      "attachmentName": "Pay Slip",
      "requiredFlag": false,
      "fileName": "712.GIF",
      "fileSize": 31133,
      "fileURL": "http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/69/41/712.GIF",
      "contentType": "image/gif",
      "timeUploaded": "2015-01-27T14:51:57+11:00"
    },
    {
      "id": 42,
      "attachmentName": "Employment Contract",
      "requiredFlag": false,
      "fileName": "ABC-123.pdf",
      "fileSize": 29595,
      "fileURL": "http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/69/42/ABC-123.pdf",
      "contentType": "application/pdf",
      "timeUploaded": "2015-01-27T14:52:00+11:00"
    }
  ]
}
```


GET Submission Data

Return the submission data for the data URL specified in the submission JSON message. The returned data will have the Content-Type disposition header set for the particular file type. Data will also be compressed if the "Accept-Encoding: gzip" header is set.

URL Examples

```
GET http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/form.xml
GET http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/receipt.pdf
GET http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/extract.xls
GET http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/15/Greenbook_Final.pdf
GET http://localhost:9080/manager/secure/rest/delivery/v1/maguire/rest-service/30/16/AQR001113Solution-Design.
docx
```

If the data cannot be resolved a HTTP 404 - Not Found status code will be returned.

POST Next Submission

Return the next submission 'Ready' for delivery of the specified client and delivery channel. This operation will change the submission delivery status to "In Progress" and will not be returned again by this operation.

URL Format

```
GET https://<tm server>/manager/secure/rest/delivery/1/<client code>/<delivery channel>/next/
```

URL Example

```
POST https://transact.maguire.com/manager/secure/rest/delivery/v1/maguire/credit-card/next/
```

200 Response

The response is the same as the GET Submission above except, the delivery status values will also be set.

```
{
  "id": 69,
  ...
  "deliveryStatus": "In Progress",
  "deliveryMethod": "REST Service",
  "deliveryMessage": "Delivered via \u0027REST Service\u0027 REST Service to administrator",
  "deliveryChannel": "REST Service",
  "deliveryTime": "2015-01-27T14:53:00+11:00"
  ...
}
```

POST Delivery Completed

Set the submission delivery to be completed, and optionally specify whether the transaction data should be deleted once delivery has been completed.

URL Format

```
POST https://<tm server>/manager/secure/rest/delivery/1/<client code>/<delivery channel>/<submission id>/
```

URL Example

```
POST https://transact.maguire.com/manager/secure/rest/delivery/v1/maguire/credit-card/69/
```

Request Body

The JSON request message must specify the transaction data SHA-256 hash values (Hex encoded) for the form XML data and the file upload attachment data.

```
{
    "xmlSHA256": "bd2f85644beffdd101be4137c68269627bbb935d8cd231ab2e5c505eabff72c6",
    "attachments": [
        {
            "id": 41,
            "fileSHA256": "3b80d18bc78aad01c50efbad9355a48a6907bdb850b96fc919570209e8cfc4c4"
        },
        {
            "id": 42,
            "fileSHA256": "553a417def543b52bcc786f1e2b79dbfd958348c94f689e8d85181aacd039a61"
        }
    ],
    "deleteData": true,
    "processingStatus": "Your application is being processed"
}
```

Delete Data Option

If the **`"deleteData": "true"`** attribute is set then JSON confirmation message then the transaction user data will be deleted immediately from the server. The delete data will include the form XML data, PDF receipt and any file attachments, but will not include the transaction meta data.

Note: you should only do this if you have pressing security requirements that requiring immedate transaction data purging after delivery. Deleted data is not recoverable from the server. Please ensure any delivered transaction data is written to HA storage systems.

Processing Status Option

If the **`processingStatus`** attribute is set then a processing status message will be added to the transaction. This message can be use to display transaction processing status updates to the end user. This text value should be 100 characters or less, longer values will be truncated.

200 Response

```
{
  "id": 69,
  ...
  "processingStatus": "Your application is being processed",
  "processingStatusTime": "2015-01-27T14:53:00+11:00",
  ...
  "deliveryStatus": "Completed",
  "deliveryMethod": "REST Service",
  "deliveryMessage": "Delivered via \u0027REST Service\u0027 REST Service to administrator",
  "deliveryChannel": "REST Service",
  "deliveryTime": "2015-01-27T14:53:00+11:00"
  ...
}
```

400 Response

If a Bad Request error occurs a JSON response message will be returned describing the error, for example:

```
400 - Invalid complete submission request, xmlSHA256 does not match stored value
```

POST Delivery Error

This operation is supported in Transaction Manager 4.3 and later.

Update the transaction delivery status to 'Error' indicating that an internal delivery error has occurred. An internal error could be when attempting to store transaction data to a disk volume which is not available.

By using this operation you can inform TM operational staff that a delivery error has occurred and this matter needs to be attended to. Staff who are in the alert group 'Receive System Alerts - Delivery' will receive an email notification when a delivery error is recored on the TM server.

If the Delivery Channel is configure to enable delivery retry, the transaction will be returned to the delivery queue once the retry delay period has been reached. If there is no delivery retry configured then submission will not be automatically returned to the delivery queue, and it will be up to an administrator to resolve the issue.

URL Format

```
POST https://<tm server>/manager/secure/rest/delivery/1/<client code>/<delivery channel>/<submission id>/
```

URL Example

```
POST https://transact.maguire.com/manager/secure/rest/delivery/v1/maguire/credit-card/2039/
```

Request Body

The JSON request message must specify an errorMessage value describing the error.

```
{
    "errorMessage": "SAN delivery storage volume not available.",
}
```

200 Response

```
{
  "id": 2039,
  ...
  "deliveryStatus": "Error",
  "deliveryMethod": "REST Service",
  "deliveryTime": "2015-01-27T14:53:00+11:00"
  ...
}
```

Form Groups API

REST Form Groups API

The Form Groups API provides a REST service for querying and creating Form Groups in Transaction Manager in a secure and efficient manner.

Security

To access the Form Groups REST endpoint the caller needs to be authenticated using HTTP Basic Authentication. The authenticated user will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission '**`REST Form Groups API`**'.

URL Endpoint

The REST Form Groups API URL endpoint is as follows:

```
https://<TM server>/manager/secure/rest/form-groups/v1/
```

Service API

This section provides a description all the REST Form Groups API operations including:

- GET Form Group
- GET Form Groups
- POST Create Form Group

GET Form Group

Retrieve a single form group by specifying the database ID.

URL Format

```
GET https://<tm server>/manager/secure/rest/form-groups/v1/<group ID>
```

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/form-groups/v1/23/
```

200 Response

```
{
    "id": 23,
    "groupName": "Reviewer Group",
    "groupDescription": "Application Reviewers Group",
    "formWorkGroup": true,
    "newForm": true,
    "savedForm": true,
    "completedForm": true,
    "taskAssign": true
}
```

GET Form Groups

Retrieve the list of form groups sorted by database ID in descending order (the ordering can be customized).

URL Format

```
GET https://<tm server>/manager/secure/rest/form-groups/v1/
```

Request Parameters

The table below shows the set of request parameters can be used as additional search criteria.

Parameter	Description
id	the unique form group id
groupName	filters on the form group name
formWorkGroup	filters on the `formWorkGroup` access control permission attribute [true | false]

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/form-groups/v1/?groupName=Card Applicants
```

200 Response

```
[
  {
    "id": 42,
    "groupName": "Card Applicants",
    "groupDescription": "Credit Card Applicants",
    "formWorkGroup": false,
    "newForm": true,
    "savedForm": true,
    "completedForm": true,
    "taskAssign": false
  }
]
```

POST Create Form Group

Create a form group that a user or task can be assigned to.

URL Format

```
POST https://<tm server>/manager/secure/rest/form-groups/v1/
```

POST Body

The POST request must contain a JSON structure describing the group. To support multiple parameters this API uses the HTTP Content Type: `application/json`

The set of task JSON attributes is listed below (all attributes are optional unless indicated otherwise):

Attribute	Type	Description
groupName	String	specifies the group name, this value must be unique (required)
groupDescription	String	specifies the group description (optional)
formWorkGroup	boolean	access control permission to share the form with work group members
newForm	boolean	access control permission allowing users to open new forms
savedForm	boolean	access control permission allowing users to open saved work group forms and group assigned form tasks
completedForm	boolean	access control permission allowing users to view completed form group submissions
taskAssign	boolean	access control permission allowing users to reassign tasks to other users

Example: Creating form group

Please ensure you specify `Content-Type: application/json`

```
POST <b>/manager/secure/rest/form-groups/v1/</b> HTTP/1.1
Host: https://transact.maguire.com
Content-Type: application/json
```

```
{
    "groupName": "Acme Applicant Group",
    "groupDescription": "Acme Corporation Applicants Group",
    "formWorkGroup": true,
    "newForm": false,
    "savedForm": true,
    "completedForm": true,
    "taskAssign": false
}
```

200 Response

```
{
  "id": 52,
  "groupName": "Acme Applicant Group",
  "groupDescription": "Acme Corporation Applicants Group",
  "formWorkGroup": true,
  "newForm": false,
  "savedForm": true,
  "completedForm": true,
  "taskAssign": false
}
```

Groovy Service Invoke

REST Groovy Service Invoke v2

The REST Groovy Service Invoke API provides support for call arbitary Groovy Services from external systems. This service endpoint can be used to perform operation such as uploading reference data for use in dynamic data or form prefill services.

Security

To enable Groovy Services to be invoked via REST API the service parameter '**`REST Invoke Enabled`**' service parameter must be enabled. By default the service parameter is false, and it must be explicitly enabled.

To access the Groovy Service Invoke REST endpoint the caller needs to be authenticated using HTTP Basic Authentication. The authenticated user will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission '**`REST Groovy Service Invoke`**'.

If the user is not a global administrator, only services belonging to the organizations assigned to the user will be accessible.

The endpoint supports HTTP GET and POST operations, including file upload MULTI-PART POST requests.

URL Endpoint

The REST service definitions URL endpoint is as follows:

```
https://<tm server>/manager/secure/rest/groovy-service-invoke/v2/
```

Service API

This section provides a description all the REST Groovy Service Invoke operations including:

- GET Groovy Service List
- POST Invoke Groovy Service

GET Groovy Service List

Retrieve the list of available services of type 'Groovy Service' sorted by service name and version.

URL Format

```
GET https://<tm server>/manager/secure/rest/groovy-service-invoke/v2/
```

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/groovy-service-invoke/v2/
```

200 Response

```
[
    {
        "serviceName": "Bulk User Loader",
        "versionNumber": 1,
        "clientCode": "maguire",
        "url": "http://localhost:9080/manager/secure/rest/groovy-service-invoke/v2/maguire/bulk-user-loader/v1/"
    },
    {
        "serviceName": "Product Catalog Loader",
        "versionNumber": 1,
        "clientCode": "maguire",
        "url": "http://localhost:9080/manager/secure/rest/groovy-service-invoke/v2/maguire/product-catalog-loader
/v1/"
    },
    {
        "serviceName": "Q1 Accounts Loader",
        "versionNumber": 1,
```

```
        "clientCode": "maguire",
        "url": "http://localhost:9080/manager/secure/rest/groovy-service-invoke/v2/maguire/q1-accounts-loader/v1
/"
    },
    {
        "serviceName": "User Profile Updater",
        "versionNumber": 1,
        "clientCode": "maguire",
        "url": "http://localhost:9080/manager/secure/rest/groovy-service-invoke/v2/maguire/user-profile-updater
/v1/"
    }
]
```

POST Invoke Groovy Service

Invoke the specified service and return the results of the Groovy Service invocation. All Groovy Service invocations are recorded in the Groovy Service Log.

Please Note: all groovy service invocations must be performed using a HTTP POST or a multi-part POST operation. This follows the REST architectural pattern of not performing state changes when making HTTP GET requests.

URL Format

The service name is normalized to avoid any URL encoding issues. Please use the service URL values provided in the GET Service List operation.

```
POST https://<tm server>/manager/secure/rest/groovy-service-invoke/v2/<normalized client code>/<normalized
service name>/<version number>/
```

URL Example

```
POST https://transact.maguire.com/manager/secure/rest/groovy-service-invoke/v2/maguire/product-catalog-loader/v1/
```

POST Body

Any request parameters should be URL form encoded in the request body. **Do Not** add request parameters to the POST URL as they will not be resolved.

Binary files can be uploaded to the services using Multipart Post. This is particularly useful for loading reference data, such as Excel or CSV data. Any uploaded files will be made available as Apache Commons FileUpload `FileItem` objects.

200 Response

The response content is that returned by the invoked Groovy Service, for example:

```
Product Catalog Loaded
```

Reference Data Example

The example Groovy Service below is loading product catalog CSV file into a client property named 'Product Catalog' belonging to the 'Maguire' Organization.

Fluent Groovy Script

The REST client makes multi-part post request with the CSV file uploaded with the request parameter name 'products'. The script gets the request parameters.products `FileItem` object. The FileItem's byte data is then convered into text using the UTF-8 charset.

With the products CSV data, the script then loads it into the database using `PropertyBuilder`. This service will create or update any existing property value with the same name.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*
import javax.servlet.http.*
import org.apache.commons.fileupload.FileItem

class FluentGroovyService {
```

```
    /*
     * Perform a groovy service invocation
     *
     * return: the result object
     */
    Object invoke(SvcDef svcDef, HttpServletRequest request, User user, Map params) {

        // Get the products file upload file item
        FileItem fileItem = (FileItem) params.products

        logger.info "loading file: " + fileItem.name

        // Convert the fileItem binary data to CSV text
        String productsCSV = new String(fileItem.get(), "UTF-8")

        // Create or update the Organizations "Product Catalog" property
        new PropertyBuilder()
            .setName("Product Catalog")
            .setValue(productsCSV)
            .setClientCode(svcDef.clientCode)
            .build()

        String msg = "Imported '" + fileItem.name + "' file into 'Product Catalog' organization property"
        logger.info msg

        return msg
    }
}
```

200 Response

When the script executes it returns the following response.

```
Imported 'product-catalog-q1.csv' file into 'Product Catalog' client property
```

Service Definitions API

REST Service Definitions API

The Service Definition API provides a REST service for querying and modifying service definitions defined in Transaction Manager in a secure and efficient manner.

Security

To access the Service Definition REST endpoint the caller needs to be authenticated using HTTP Basic Authentication. The authenticated user will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission '**`REST Service Definitions API`**'.

If the user is not a global administrator, only services belonging to the organizations assigned to the user will be accessible.

URL Endpoint

The REST service definitions URL endpoint is as follows:

```
https://<TM server>/manager/secure/rest/service-definitions/v1/
```

Accessible Service Types

The REST Service Definitions service allows you to access service of the following service types only:

- Delivery Process
- Dynamic Data
- Form Prefill
- Form Saved Processor
- Form Version Selector
- Groovy Service
- Job Action
- Job Controller
- Receipt Number
- Submission Completed Processor
- Submission Data Validator
- Submission Preprocessor
- Tracking Number

Service API

This section provides a description all the REST Service Definition API operations including:

GET Service Definition List

Return the list of service definitions ordered by service name and version number.

URL Format

```
GET https://<tm server>/manager/secure/rest/service-definitions/v1/
```

Request Parameters

The table below shows the set of request parameters can be used as additional search criteria.
If multiple parameters are specified, all of them are applied to the result set.

Parameter	Description
activeFlag	a boolean parameter that filters on the active flag (Values: true, false)

clientName	if set, only services for the given organization (by name) will be returned
id	a numeric parameter containing the service definition's database ID
jobTemplateFlag	a boolean parameter that filters on the job template flag (Values: true, false)
serviceName	the service name to match (note: the combination of service name and version number uniquely identifies a service)
serviceType	the service type to match (see Accessible Service Types for the set of accessible types)
unitTestEnabledFlag	a boolean parameter that filters on the unit test enabled flag (Values: true, false)
versionNumber	a numeric value containing the version number to match (note: the combination of service name and version number uniquely identifies a service)

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/service-definitions/v1/?serviceType=Form%20Version%
20Selector&activeFlag=true
```

200 Response


```
[
  {
    "id": 1226,
    "serviceName": "A/B Testing Form Version Selector",
    "versionNumber": 1,
    "serviceType": "Form Version Selector",
    "description": "A service that selects the form version to render at random from the list of form versions",
    "serviceTypeDefaultFlag": false,
    "activeFlag": true,
    "jobTemplateFlag": false,
    "unitTestEnabledFlag": false,
    "classnameBeanname": "com.avoka.fc.core.service.form.RandomFormVersionSelectorService",
    "createdAt": "Jun 13, 2014 10:20:15 AM",
    "createdBy": "system",
    "lastModifiedAt": "Mar 4, 2015 3:58:17 AM",
    "lastModifiedBy": "medgar",
    "serviceParameters": [
      {
        "id": 1931,
        "name": "includedFormVersions",
        "description": "A comma separated list of form version numbers to include in selection. Leave blank to
use all versions.",
        "type": "String",
        "bindParameterFlag": true,
        "unitTestFlag": false,
        "value": "3.0, 2.0",
        "createdAt": "Jun 13, 2014 10:20:15 AM",
        "createdBy": "system",
        "lastModifiedAt": "Mar 4, 2015 3:58:17 AM",
        "lastModifiedBy": "medgar"
      },
      {
        "id": 1932,
        "name": "ignoreFormVersionRequestParameter",
        "description": "Enable this parameter ignore the value of the 'tmFormVersion' request parameter.",
        "type": "boolean",
        "bindParameterFlag": true,
        "unitTestFlag": false,
        "value": "false",
        "createdAt": "Jun 13, 2014 10:20:15 AM",
        "createdBy": "system"
      }
    ]
  },
  {
    "id": 1206,
    "serviceName": "Current Version Selector",
    "versionNumber": 1,
    "serviceType": "Form Version Selector",
    "description": "This service returns the current form version, or (if specified) uses the request parameter
'tmFormVersion' to look up a specific version.",
    "serviceTypeDefaultFlag": true,
    "activeFlag": true,
    "jobTemplateFlag": false,
    "unitTestEnabledFlag": false,
```

```
    "classnameBeanname": "com.avoka.fc.core.service.form.CurrentVersionSelectorService",
    "createdAt": "Jun 11, 2014 3:13:31 PM",
    "createdBy": "system",
    "lastModifiedAt": "Feb 11, 2015 3:12:49 PM",
    "lastModifiedBy": "someadmin",
    "serviceParameters": [
      {
        "id": 1911,
        "name": "ignoreFormVersionRequestParameter",
        "description": "Enable this parameter to always use the current form version, regardless of the value of
the 'tmFormVersion' request parameter.",
        "type": "boolean",
        "bindParameterFlag": true,
        "unitTestFlag": false,
        "value": "false",
        "createdAt": "Jun 11, 2014 3:13:31 PM",
        "createdBy": "system"
      }
    ]
  }
]
```

GET Service Definition

Return the service definition for the specified service id.

URL Format

```
https://<tm server>/manager/secure/rest/service-definitions/v1/<service ID>/
```

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/service-definitions/v1/803/
```

200 Response

```
{
  "id": 803,
  "serviceName": "Random Tracking Number",
  "versionNumber": 1,
  "serviceType": "Tracking Number",
  "description": "Create random form transaction tracking numbers.",
  "serviceTypeDefaultFlag": true,
  "activeFlag": true,
  "jobTemplateFlag": false,
  "unitTestEnabledFlag": false,
  "classnameBeanname": "com.avoka.fc.core.service.form.RandomTrackingNumberService",
  "createdAt": "Feb 27, 2014 5:14:18 PM",
  "createdBy": "system",
  "lastModifiedAt": "Aug 12, 2014 3:14:28 PM",
  "lastModifiedBy": "someadmin",
  "serviceParameters": [
    {
      "id": 1348,
      "name": "numberLength",
      "description": "The length of the tracking number.",
      "type": "List",
      "bindParameterFlag": true,
      "unitTestFlag": false,
      "listValues": "4:4|5:5|6:6|7:7|8:8",
      "value": "6",
      "createdAt": "Feb 27, 2014 5:14:18 PM",
      "createdBy": "system"
    },
    {
      "id": 1349,
      "name": "characterValues",
      "description": "The list of character values to generate the random tracking number from.",
      "type": "String",
      "bindParameterFlag": true,
      "unitTestFlag": false,
```

```
      "value": "ABCDEFGHJKLMNPQRSTVWXYZ23456789",
      "createdAt": "Feb 27, 2014 5:14:18 PM",
      "createdBy": "system",
      "lastModifiedAt": "Aug 12, 2014 3:14:28 PM",
      "lastModifiedBy": "someadmin"
    }
  ]
}
```

GET Service Definition Parameter List

Returns all service parameters for the specified service definition, sorted by parameter name.

URL Format

```
https://<tm server>/manager/secure/rest/service-definitions/v1/<service ID>/serviceParameters
```

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/service-definitions/v1/803/serviceParameters
```

200 Response

```
[
  {
    "id": 1349,
    "name": "characterValues",
    "description": "The list of character values to generate the random tracking number from.",
    "type": "String",
    "bindParameterFlag": true,
    "unitTestFlag": false,
    "value": "ABCDEFGHJKLMNPQRSTVWXYZ23456789",
    "createdAt": "Feb 27, 2014 5:14:18 PM",
    "createdBy": "system",
    "lastModifiedAt": "Aug 12, 2014 3:14:28 PM",
    "lastModifiedBy": "someadmin"
  },
  {
    "id": 1348,
    "name": "numberLength",
    "description": "The length of the tracking number.",
    "type": "List",
    "bindParameterFlag": true,
    "unitTestFlag": false,
    "listValues": "4:4|5:5|6:6|7:7|8:8",
    "value": "6",
    "createdAt": "Feb 27, 2014 5:14:18 PM",
    "createdBy": "system"
  }
]
```

GET Service Definition Parameter

Return the service parameter for the specified service definition id and parameters id.

URL Format

```
https://<tm server>/manager/secure/rest/service-definitions/v1/<service ID>/serviceParameters/<service parameter
ID>
```

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/service-definitions/v1/803/serviceParameters/1348
```

200 Response


```
{
  "id": 1348,
  "name": "numberLength",
  "description": "The length of the tracking number.",
  "type": "List",
  "bindParameterFlag": true,
  "unitTestFlag": false,
  "listValues": "4:4|5:5|6:6|7:7|8:8",
  "value": "6",
  "createdAt": "Feb 27, 2014 5:14:18 PM",
  "createdBy": "system"
}
```


POST Create Service Definition

Post a JSON object describing a service definition to create a service definition.

URL Format

```
https://<tm server>/manager/secure/rest/service-definitions/v1/
```

POST Body Format

You must pass in a service definition value object containing the new values.

POST Example

```
POST https://transact.maguire.com/manager/secure/rest/service-definitions/v1/
```


```
{
    "serviceName": "Random Tracking Number",
    "versionNumber": 2,
    "serviceType": "Tracking Number",
    "description": "A new version of the random tracking number service",
    "serviceTypeDefaultFlag": false,
    "activeFlag": true,
    "jobTemplateFlag": false,
    "unitTestEnabledFlag": false,
    "clientName": "Maguire",
    "classnameBeanname": "com.avoka.fc.core.service.form.RandomTrackingNumberService",
    "serviceParameters": [
      {
        "name": "numberLength",
        "description": "The length of the tracking number.",
        "type": "List",
        "bindParameterFlag": true,
        "unitTestFlag": false,
        "listValues": "4:4|5:5|6:6|7:7|8:8",
        "value": "6"
      },
      {
        "name": "characterValues",
        "description": "The list of character values to generate the random tracking number from.",
        "type": "String",
        "bindParameterFlag": true,
        "unitTestFlag": false,
        "value": "ABCDEFGHJKLMNPQRSTVWXYZ23456789"
      }
    ]
}
```


200 Response


```
{
  "id": 215,
  "serviceName": "Random Tracking Number",
  "versionNumber": 2,
```

```
  "serviceType": "Tracking Number",
  "description": "A new version of the random tracking number service",
  "serviceTypeDefaultFlag": false,
  "activeFlag": true,
  "jobTemplateFlag": false,
  "unitTestEnabledFlag": false,
  "clientName": "Maguire",
  "classnameBeanname": "com.avoka.fc.core.service.form.RandomTrackingNumberService",
  "createdAt": "Sep 30, 2015 4:08:51 PM",
  "createdBy": "someadmin",
  "serviceParameters": [
    {
      "id": 950,
      "name": "numberLength",
      "description": "The length of the tracking number.",
      "type": "List",
      "bindParameterFlag": true,
      "unitTestFlag": false,
      "listValues": "4:4|5:5|6:6|7:7|8:8",
      "value": "6",
      "createdAt": "Sep 30, 2015 4:08:51 PM",
      "createdBy": "someadmin"
    },
    {
      "id": 949,
      "name": "characterValues",
      "description": "The list of character values to generate the random tracking number from.",
      "type": "String",
      "bindParameterFlag": true,
      "unitTestFlag": false,
      "value": "ABCDEFGHJKLMNPQRSTVWXYZ23456789",
      "createdAt": "Sep 30, 2015 4:08:51 PM",
      "createdBy": "someadmin"
    }
  ]
}
```


POST Create Service Parameter

Post a JSON object describing a service parameter to create a service parameter for an existing service definition.

URL Format

```
https://<tm server>/manager/secure/rest/service-definitions/v1/<service ID>/serviceParameters
```

POST Body Format

You must pass in a service parameter value object containing the new values.

POST Example

```
POST https://transact.maguire.com/manager/secure/rest/service-definitions/v1/215/serviceParameters
```


```
{
    "name": "myCustomParameter",
    "description": "A new parameter that our service can use",
    "type": "String",
    "bindParameterFlag": false,
    "unitTestFlag": false,
    "value": "Default Value"
}
```


200 Response


```
{
  "id": 953,
  "name": "myCustomParameter",
  "description": "A new parameter that our service can use",
  "type": "String",
  "bindParameterFlag": false,
```

```
  "unitTestFlag": false,
  "value": "Default Value",
  "createdAt": "Oct 1, 2015 11:26:08 AM",
  "createdBy": "someadmin"
}
```

PUT Update Service Definition

Put a JSON object describing an existing service definition to update its values.

URL Format

```
https://<tm server>/manager/secure/rest/service-definitions/v1/<service ID>
```

PUT Body Format

You must pass in a service definition value object containing the new values.

PUT Example

```
PUT https://transact.maguire.com/manager/secure/rest/service-definitions/v1/215
```

```
{
    "serviceName": "My Random Tracking Number",
    "versionNumber": 3,
    "serviceType": "Tracking Number",
    "description": "An updated version of the random tracking number service",
    "serviceTypeDefaultFlag": false,
    "activeFlag": true,
    "serviceParameters": [
      {
        "name": "numberLength",
        "description": "The length of the tracking number.",
        "type": "List",
        "bindParameterFlag": true,
        "unitTestFlag": false,
        "listValues": "4:4|5:5|6:6|7:7|8:8",
        "value": "8"
      },
      {
        "name": "characterValues",
        "description": "The list of character values to generate the random tracking number from.",
        "type": "String",
        "bindParameterFlag": true,
        "unitTestFlag": false,
        "value": "AEIOU"
      }
    ]
}
```

200 Response

```
{
  "id": 215,
  "serviceName": "My Random Tracking Number",
  "versionNumber": 3,
  "serviceType": "Tracking Number",
  "description": "An updated version of the random tracking number service",
  "serviceTypeDefaultFlag": false,
  "activeFlag": true,
  "jobTemplateFlag": false,
  "unitTestEnabledFlag": false,
  "clientName": "Maguire",
  "classnameBeanname": "com.avoka.fc.core.service.form.RandomTrackingNumberService",
  "createdAt": "Sep 30, 2015 4:08:51 PM",
  "createdBy": "someadmin",
  "lastModifiedAt": "Oct 1, 2015 10:27:19 AM",
  "lastModifiedBy": "someadmin",
  "serviceParameters": [
    {
```

```
      "id": 949,
      "name": "characterValues",
      "description": "The list of character values to generate the random tracking number from.",
      "type": "String",
      "bindParameterFlag": true,
      "unitTestFlag": false,
      "value": "AEIOU",
      "createdAt": "Sep 30, 2015 4:08:51 PM",
      "createdBy": "someadmin",
      "lastModifiedAt": "Oct 1, 2015 10:27:19 AM",
      "lastModifiedBy": "someadmin"
    },
    {
      "id": 950,
      "name": "numberLength",
      "description": "The length of the tracking number.",
      "type": "List",
      "bindParameterFlag": true,
      "unitTestFlag": false,
      "listValues": "4:4|5:5|6:6|7:7|8:8",
      "value": "8",
      "createdAt": "Sep 30, 2015 4:08:51 PM",
      "createdBy": "someadmin",
      "lastModifiedAt": "Oct 1, 2015 10:27:19 AM",
      "lastModifiedBy": "someadmin"
    }
  ]
}
```


PUT Update Service Parameter

Put a JSON object describing a service parameter to update an existing service parameter.

URL Format

```
https://<tm server>/manager/secure/rest/service-definitions/v1/<service ID>/serviceParameters/<service parameter ID>
```

PUT Body Format

You must pass in a service parameter value object containing the new values.

PUT Example

```
PUT https://transact.maguire.com/manager/secure/rest/service-definitions/v1/215/serviceParameters/953
```


```
{
  "name": "myCustomParameter2",
  "description": "An updated parameter that our service can use",
  "type": "String",
  "bindParameterFlag": false,
  "unitTestFlag": false,
  "value": "Updated Value"
}
```


200 Response


```
{
  "id": 953,
  "name": "myCustomParameter2",
  "description": "An updated parameter that our service can use",
  "type": "String",
  "bindParameterFlag": false,
  "unitTestFlag": false,
  "value": "Updated Value",
  "createdAt": "Oct 1, 2015 11:26:08 AM",
  "createdBy": "someadmin",
  "lastModifiedAt": "Oct 1, 2015 11:35:22 AM",
  "lastModifiedBy": "someadmin"
}
```

DELETE Service Definition

Irrevocably delete a service definition and all its parameters. **USE THIS OPERATION WITH CAUTION**.

URL Format

```
https://<tm server>/manager/secure/rest/service-definitions/v1/<service ID>
```

or

```
https://<tm server>/manager/secure/rest/service-definitions/v1/<service name>/<version number>
```

DELETE Examples

```
DELETE https://transact.maguire.com/manager/secure/rest/service-definitions/v1/215
```

and

```
DELETE https://transact.maguire.com/manager/secure/rest/service-definitions/v1/My Random Tracking Number/3
```

200 Response

The response will not contain any additional information. If the status code is 200, the service definition was deleted successfully.

DELETE Service Parameter

Irrevocably delete a service parameter. **USE THIS OPERATION WITH CAUTION**.

URL Format

```
https://<tm server>/manager/secure/rest/service-definitions/v1/<service ID>/serviceParameters/<service parameter ID>
```

DELETE Example

```
DELETE https://transact.maguire.com/manager/secure/rest/service-definitions/v1/215/serviceParameters/953
```

200 Response

The response will contain the updated service definition and the remaining parameters.


```
{
  "id": 215,
  "serviceName": "My Random Tracking Number",
  "versionNumber": 3,
  "serviceType": "Tracking Number",
  "description": "An updated version of the random tracking number service",
  "serviceTypeDefaultFlag": false,
  "activeFlag": true,
  "jobTemplateFlag": false,
  "unitTestEnabledFlag": false,
  "clientName": "Maguire",
  "classnameBeanname": "com.avoka.fc.core.service.form.RandomTrackingNumberService",
  "createdAt": "Sep 30, 2015 4:08:51 PM",
  "createdBy": "someadmin",
  "lastModifiedAt": "Oct 1, 2015 11:43:21 AM",
  "lastModifiedBy": "someadmin",
  "serviceParameters": [
    {
      "id": 949,
      "name": "characterValues",
      "description": "The list of character values to generate the random tracking number from.",
```

```
      "type": "String",
      "bindParameterFlag": true,
      "unitTestFlag": false,
      "value": "AEIOU",
      "createdAt": "Sep 30, 2015 4:08:51 PM",
      "createdBy": "someadmin",
      "lastModifiedAt": "Oct 1, 2015 10:27:19 AM",
      "lastModifiedBy": "someadmin"
    },
    {
      "id": 950,
      "name": "numberLength",
      "description": "The length of the tracking number.",
      "type": "List",
      "bindParameterFlag": true,
      "unitTestFlag": false,
      "listValues": "4:4|5:5|6:6|7:7|8:8",
      "value": "8",
      "createdAt": "Sep 30, 2015 4:08:51 PM",
      "createdBy": "someadmin",
      "lastModifiedAt": "Oct 1, 2015 10:27:19 AM",
      "lastModifiedBy": "someadmin"
    }
  ]
}
```

POST Upload Service Archive

Upload a Service Archive ZIP file into the TM server creating or updating any included service definitions.

Please note this will preserve any existing global default type Services and existing Service Connections. However any existing services of the same name will be updated with the provided service archive.

URL Format

```
POST https://<tm server>/manager/secure/rest/service-definitions/v1/upload-service-archive/
```

POST Body

The Multipart POST request parameters include:

- **`archiveFile`** - the service archive ZIP file (required)
- **`clientCode`** - the Organization client code (optional), if not specified the current client will be used

Example

Please note POST example below is idealized to provide an illustration of multipart POST request.

```
POST <b>/manager/secure/rest/service-definitions/v1/upload-service-archive/</b> HTTP/1.1
Host: https://transact.maguire.com
Content-type: multipart/form-data, boundary=AaB03x

--AaB03x
content-disposition: form-data; name="archiveFile"; filename="service-archive-Groovy_Form_Security_Filter-2016-
06-17.zip"
Content-Transfer-Encoding: binary
...
--AaB03x--
==========================
```

200 Response

```
{
    "archiveName": "service-archive-Groovy_Form_Security_Filter-2016-06-17.zip",
    "importMessage": "Imported Groovy Form Security Filter - v1 successfully.",
    "importStatus": "Completed",
    "importTime": "2016-06-17T16:47+1000"
}
```

POST Run Unit Test

Run the service's automated Unit Test.

URL Format

```
POST https://<tm server>/manager/secure/rest/service-definitions/v1/run-unit-test/
```

POST Body

The POST request parameters include:

- **`serviceName`** - the name of the service definition to run (required)
- **`versionNumber`** - the service version number (optional), highly recommended to resolve correct version
- **`clientCode`** - the Organization client code (optional), highly recommended to resolve correct Organization service

Example

This example below runs the unit test for the 'Hello World' version 1 service. The example the message body below formatted for easier reading.

Please ensure you specify **`Content-Type: application/x-www-form-urlencoded`**

```
POST <b>/manager/secure/rest/service-definitions/v1/run-unit-test/</b> HTTP/1.1
Host: https://transact.maguire.com
Content-Type: application/x-www-form-urlencoded

<b>serviceName</b>=Hello World&<b>versionNumber</b>=1&<b>clientCode</b>=maguire
```

200 Response

Below is a example response of a service unit test success.

```
{
  "serviceName": "Hello World",
  "versionNumber": 1,
  "testStatus": "Success",
  "message": "Test succeeded in 49 ms",
  "logger": "21:24:21,034 INFO  groovy result"
}
```

Below is a example response of a service unit test failure.

```
{
  "serviceName": "Hello World",
  "versionNumber": 1,
  "testStatus": "Failure",
  "message": "Test failed after 741 ms",
  "errorCause": "java.lang.NullPointerException",
  "errorLine": 19
}
```

Service Definition and Service Parameter JSON

The POST and PUT operations described above work with service definition and parameter JSON value objects.
See the previous sections for a number of examples.

Service Definition

Attribute	Type	Description
activeFlag	Boolean	a flag describing whether the service is active and usable
classnameBean name	String	the Java class name/bean name of the class implementing the service
clientName	String	the name of the organization the service is associated with Note: This value cannot be changed once a service has been created.
createdAt	Date	the service creation date Note: This parameter is ignored during updates as TM maintains these attributes.

createdBy	String	the user who created the service Note: This parameter is ignored during updates as TM maintains these attributes.
description	String	a description of the service that will be visible to administrators
id	Long	the database ID identifying the service Note: You should not need to set this in your POST and PUT calls. However, it is important to use the IDs returned by a GET call, as the service ID is used in URLs. However, do not use the ID across sessions as database IDs may change (e.g. because services may get recreated, or DB backups restored)
jobTemplateFlag	Boolean	a flag describing whether the service contains a job template
lastModifiedAt	Date	the service last modified date Note: This parameter is ignored during updates as TM maintains these attributes.
lastModifiedBy	String	the user who last modified the service Note: This parameter is ignored during updates as TM maintains these attributes.
serviceName	String	the name identifying the service (required for new services) Note: Name and version number together uniquely identify a service.
serviceParameters	List of Service Parameter	the list of service parameter value objects
serviceType	String	the service type, e.g. Dynamic Data Note: This value is required for new services but cannot be changed later.
serviceTypeDefaultFlag	Boolean	a flag describing whether the service is the default service for its type
unitTestEnabledFlag	Boolean	a flag describing whether the service is set up for unit testing
versionNumber	Integer	the version number, e.g. 1 (required for new services) Note: Name and version number together uniquely identify a service.

Service Parameter

Attribute	Type	Description
bindParameterFlag	Boolean	a flag describing whether the service parameter name corresponds to a property on the service class and should be set automatically by TM Note: If you use this flag incorrectly, your service will not work. Ensure to turn this flag on if and only if the service class has a setter method corresponding to the parameter name.
createdAt	Date	the service parameter creation date Note: This parameter is ignored during updates as TM maintains these attributes.
createdBy	String	the user who created the service parameter Note: This parameter is ignored during updates as TM maintains these attributes.
description	String	a description of the service parameter that will be visible to administrators
id	Long	the database ID identifying the service parameter Note: You should not need to set this in your POST and PUT calls. However, it is important to use the IDs returned by a GET call, as the service parameter ID is used in URLs. However, do not use the ID across sessions as database IDs may change (e.g. because services may get recreated, or DB backups restored)
lastModifiedAt	Date	the service parameter last modified date Note: This parameter is ignored during updates as TM maintains these attributes.
lastModifiedBy	String	the user who last modified the service parameter Note: This parameter is ignored during updates as TM maintains these attributes.
listValues	String	for service parameters of type "List", this parameter contains the possible values that can be assigned to this parameter the format is pipe-separated with optional display names: value:displayName|anotherValue|thirdValue:thirdDisplayName example for a parameter controlling expiry: "0:Never|1:1 day|3:3 days|7:1 week|365:1 year"
name	String	the name identifying the service parameter (required) Note: Parameter names are unique within a service.
unitTestFlag	Boolean	a flag describing whether the service parameter is used for unit testing
type	String	the type of values that can be assigned to the service parameter This can be one of: • Boolean: Possible values "true", "false" • Date: Date/time values • Email: Valid email address • Groovy Script: Basically a string, but TM provides a specialized Groovy editor • HTML • JSON • List: The value is a string, but the set of possible values can be defined in the "listValues" parameter • Number • Password: A password (TM will not display the value)

		• String
value	String	the value assigned to the service parameter Note: The value must conform to the service parameter type.

Tasks API

REST Tasks API

The Tasks API provides a REST service for creating and updating tasks in Transaction Manager in a secure and efficient manner.

Security

To access the Tasks REST endpoint the caller needs to be authenticated using HTTP Basic Authentication. The authenticated user will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission '**`REST Tasks API`**'. You will also need to be either an administrator with global access or be associated with an organization to access its transaction records.

URL Endpoint

The REST Tasks API URL endpoint is as follows:

```
https://<TM server>/manager/secure/rest/tasks/v1/
```

Service API

This section provides a description all the REST Tasks API operations including:

- POST Create Task
- PUT Update Task

POST Create Task

Create a task that can be assigned to a user or a group.

URL Format

```
POST https://<tm server>/manager/secure/rest/tasks/v1/
```

POST Body

The POST request must contain a request parameter named "task", whose value contains a JSON structure describing the task. To support multiple parameters this API uses the HTTP Content Type: **`application/x-www-form-urlencoded`**

The set of task JSON attributes is listed below (all attributes are optional unless indicated otherwise):

Attribute	Type	Description
allowClaimFlag	boolean	a flag describing whether the task will be claimable (relevant for tasks assigned to groups)
address	String	a string containing the physical address related to the task
contactEmailAddress	String	the email address to send email to (must be specified if sendEmailFlag is set to true)
datetimeExpiry	String	a JSON ISO DateTime string without milliseconds describing when the task should expire automatically if it has not been completed the required date format is yyyy-MM-dd'T'HH:mm:ssZZ
datetimeScheduled	String	a JSON ISO DateTime string without milliseconds describing when the task is scheduled the required date format is yyyy-MM-dd'T'HH:mm:ssZZ
emailMessage	String	the email message to use in the task notification email (used only if sendEmailFlag is set to true)
emailSubject	String	the email subject to use in the task notification email (used only if sendEmailFlag is set to true)
formCode	String	the form code of the form that the task will be based on (required)
formPrefillServiceName	String	the name of a service definition of type 'Form Prefill' that will be used to provide prefill XML data at task creation time (used only if taskType is "Anonymous")
groups	List of String	A list of group names to which the task shall be assigned (required if userLoginName is not set and taskType is not "Anonymous")
latitude	Double	The latitude of the location the task relates to
longitude	Double	The longitude of the location the task relates to
portalName	String	The name of the space that will be hosting the task (required if taskType is set to "Anonymous")
receiptNumber	String	The receipt number that will be used for the task transaction

reviewSubmissionId	String	The database ID of the submission that this task will review (required and used only if taskType is set to "Review")
saveChallengeAnswer	String	If you would like to secure the task further, you can specify a secret answer the user will have to enter to access the form (used only if taskType is set to "Anonymous")
sendEmailFlag	boolean	a flag controlling whether an email shall be sent to notify of the new task (if taskType is set to "Anonymous", the email will be sent to the contact email address)
sequence	Integer	The task sequence number (mainly used in collaboration job task sequencing)
taskMessage	String	The task message that will be displayed to assignees
taskSubject	String	The task subject that will be displayed to assignees
taskType	String	The type of task you want to create (required). Possible values: • Anonymous: Create an anonymous saved form • Form: Create a standalone task assigned to a user or one or more groups • Review: Create a task relating to a previous submission and assign it to a user or one or more groups
transRefNumber	String	A custom reference number that will be stored in the transaction object (used only if the taskType is set to "Anonymous")
userDeletableFlag	boolean	a flag controlling whether the task can be deleted by assignees
userLoginName	String	The login name of the assignee (used only if taskType is not "Anonymous")

In addition to the **`task`** request parameter, there are two optional parameters you can use to pass in XML task data:

- **`formDataXml`** - the XML data to be used as the seed file
- **`inputXmlData`** - the XML data that will be mapped into the form using XML prefill mappings set up in TM

The two XML parameters can be passed in as request parameters, or alternatively can be included in a Multipart POST request.

Example: Creating an anonymous saved form

This example creates an anonymous saved form secured with a secret code. The person who will fill in the form is notified via email. The example the message body below formatted for easier reading.

Please ensure you specify **`Content-Type: application/x-www-form-urlencoded`**

```
POST <b>/manager/secure/rest/tasks/v1/</b> HTTP/1.1
Host: https://transact.maguire.com
Content-Type: application/x-www-form-urlencoded

<b>task</b>={
    "taskType": "Anonymous",
    "portalName": "Maguire",
    "formCode": "pf-lt41",
    "taskSubject": "Please complete this submission",
    "taskMessage": "This submission has been assigned to you to complete.",
    "saveChallengeAnswer": "b3d5",
    "contactEmailAddress": "test@avoka.com",
    "sendEmailFlag": true,
    "formPrefillServiceName": "Task Prefill"
}
```

200 Response

```
{
  "id": 73003,
  "trackingCode": "7FAE76",
  "submitKey": "01b2b3c0ac09dc706fb81c0e47731724",
  "taskFlag": false,
  "taskCreatedTimestamp": "2015-10-06T14:04:40+11:00",
  "taskSubject": "Please complete this submission",
  "taskMessage": "This submission has been assigned to you to complete.",
  "formCode": "pf-lt41",
  "formName": "Payment Form",
  "formVersion": "1.0",
  "clientCode": "lt41",
  "clientName": "Test Client (4.1)",
  "portalName": "Maguire",
  "fieldWorkerFlag": false,
  "testMode": false,
  "contactEmailAddress": "test@avoka.com",
  "formStatus": "Saved",
  "timeSubmission": "2015-10-06T14:04:39+11:00",
  "userAgentBrowser": "",
  "userAgentOs": "",
```

```
  "deliveryStatus": "Not Ready",
  "dataDeleted": false,
  "formURL": "https://transact.maguire.com/web-plugin/servlet/SmartForm.html?
submitKey=01b2b3c0ac09dc706fb81c0e47731724"
}
```

Example: Creating a form task with XML form data

This example creates a standalone form task assigned to a single user with the **`task`** parameter, and includes the XML form data provided with the **`formDataXml`** parameter.

Please note the form prefill data is **`application/x-www-form-urlencoded`** encoded, and the message body below formatted for easier reading. In practice it may be eaiser to use a Multipart POST request to upload the form XML data.

```
POST <b>/manager/secure/rest/tasks/v1/</b> HTTP/1.1
Host: https://transact.maguire.com
Content-Type: application/x-www-form-urlencoded

<b>task</b>={
    "taskType": "Form",
    "formCode": "pf-lt41",
    "taskSubject": "Monthly activity summary",
    "taskMessage": "Please fill in the monthly activity summary for 2015-09",
    "transRefNumber": "ma-201509-jdoe",
    "sendEmailFlag": true,
    "emailSubject": "TASK: Monthly activity summary",
    "datetimeScheduled": "2015-10-09T09:00:00%2b10:00",
    "datetimeExpiry": "2015-10-16T23:59:59%2b10:00",
    "userLoginName": jdoe,
    "userDeletableFlag": false
}
&<b>formDataXml</b>=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22%3F%3E%0A%3CAvokaSmartForm%3E%0A++%
3CAddressAustralia%3E%0A++++%3CAddressLine1%2F...
```

200 Response

```
{
  "id": 73004,
  "trackingCode": "3ZYAA3",
  "submitKey": "d97d8a4be640d6a04cec039a1a657971",
  "taskFlag": true,
  "taskKey": "d97d8a4be640d6a04cec039a1a657971",
  "taskType": "Form",
  "taskCreatedTimestamp": "2015-10-06T14:09:31+11:00",
  "taskScheduledTimestamp": "2015-10-09T10:00:00+11:00",
  "taskExpiryTimestamp": "2015-10-17T00:59:59+11:00",
  "taskSubject": "Monthly activity summary",
  "taskMessage": "Please fill in the monthly activity summary for 2015-09",
  "formCode": "pf-lt41",
  "formName": "Payment Form",
  "formVersion": "1.0",
  "clientCode": "lt41",
  "clientName": "Test Client (4.1)",
  "fieldWorkerFlag": false,
  "testMode": false,
  "userLoginName": "jdoe",
  "formStatus": "Assigned",
  "userAgentBrowser": "",
  "userAgentOs": "",
  "dataDeleted": false
}
```

Example: Creating a review task

This example creates a review task assigned to a group of users.

```
POST <b>/manager/secure/rest/tasks/v1/</b> HTTP/1.1
Host: https://transact.maguire.com
Content-Type: application/x-www-form-urlencoded

<b>task</b>={
```

```
    "taskType": "Review",
    "formCode": "pf-lt41",
    "taskSubject": "Please review application no. 69618",
    "sendEmailFlag": false,
    "address": "The Corso, Manly NSW 2095",
    "latitude": -33.86,
    "longitude": 151.2,
    "userDeletableFlag": false,
    "allowClaimFlag": true,
    "groups": [
        "Job Reviewers",
        "Job Managers"
    ],
    "reviewSubmissionId": "69618"
}
```

200 Response

```
{
  "id": 73008,
  "trackingCode": "KVGGBB",
  "submitKey": "13537f9ab228a8bc50eae0e5d8bb3436",
  "taskFlag": true,
  "taskKey": "13537f9ab228a8bc50eae0e5d8bb3436",
  "taskType": "Review",
  "taskCreatedTimestamp": "2015-10-06T14:14:49+11:00",
  "taskSubject": "Please review application no. 69618",
  "taskAddress": "The Corso, Manly NSW 2095",
  "taskLatitude": -33.86,
  "taskLongitude": 151.2,
  "formCode": "pf-lt41",
  "formName": "Payment Form",
  "formVersion": "1.0",
  "clientCode": "lt41",
  "clientName": "Test Client (4.1)",
  "fieldWorkerFlag": false,
  "testMode": false,
  "submissionGroupNames": "Job Managers, Job Reviewers",
  "formStatus": "Assigned",
  "userAgentBrowser": "",
  "userAgentOs": "",
  "attachmentsStatus": "Optional",
  "dataDeleted": false
}
```

PUT Update Task

Update the task's form XML data and/or status.

URL Format

```
PUT https://<tm server>/manager/secure/rest/tasks/v1/<transaction id>
```

PUT Body

The body of your PUT request must one or more of the following parameters:

- **`formDataXml`** - the form XML data to update the task with
- **`formStatus`** - the form status information to update the task with. Must be either "Assigned", "Saved" or "Abandoned".
- **`processingStatus`** - the transaction processing status entry to add (visible to the user), but does not otherwise influence transaction processing.

Example: Updating a Task

The example below updates Task ID 72492 (as specified by the URL path transaction id), with the form XML data, form status and processing status message.

Please note the form prefill data is **`application/x-www-form-urlencoded`** encoded, and the message body below formatted for easier reading. In practice it may be eaiser to use a Multipart POST request to upload the form XML data.

```
POST <b>/manager/secure/rest/tasks/v1/72492/</b> HTTP/1.1
Host: https://transact.maguire.com
Content-Type: application/x-www-form-urlencoded

<b>formDataXml</b>=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22%3F%3E%0A%3CAvokaSmartForm%3E%0A++%
3CAddressAustralia%3E%0A++++%3CAddressLine1%2F...
&<b>formStatus</b>=Saved
&<b>processingStatus</b>=Reverted to saved status at user request
```

200 Response


```
{
  "id": 72492,
  "trackingCode": "P74L3V",
  "receiptNumber": "pf-lt41-16",
  "submitKey": "9f746b51721051b674b057c37b4bee95",
  "taskFlag": false,
  "formCode": "pf-lt41",
  "formName": "Payment Form",
  "formVersion": "1.0",
  "clientCode": "lt41",
  "clientName": "Test Client (4.1)",
  "portalName": "Maguire",
  "fieldWorkerFlag": false,
  "testMode": false,
  "userLoginName": "john.smith",
  "formStatus": "Saved",
  "timeRequest": "2015-10-02T14:07:39+10:00",
  "timeSubmission": "2015-10-02T14:10:25+10:00",
  "timeLastUserActivity": "2015-10-02T14:11:14+10:00",
  "timeToSubmitSec": 166,
  "ipAddress": "123.134.145.156",
  "requestCookie": "JSESSIONID=pfD+pEOezdTwyev-jINZSn1y; __utma=264897119.59525688.1351658449.1393997213...",
  "referer": "https://transact.maguire.com/maguire/login.htm",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.0",
  "userAgentBrowser": "Firefox 40",
  "userAgentBrowserType": "Firefox",
  "userAgentBrowserVersion": "40",
  "userAgentDeviceType": "Desktop",
  "userAgentOs": "Windows 7",
  "userAgentOsType": "Windows",
  "userAgentOsVersion": "7",
  "attachmentsStatus": "Optional",
  "paymentStatus": "Required",
  "paymentType": "Hosted",
  "paymentTotal": 123.0,
  "processingStatus": "Reverted to saved status at user request",
  "processingStatusTime": "2015-10-02T14:11:14+10:00",
  "deliveryStatus": "Not Ready",
  "dataDeleted": false,
  "formURL": "https://transact.maguire.com/workspace/secure/servlet/SmartForm.html?
submitKey=9f746b51721051b674b057c37b4bee95"
}
```

TPac API

REST TPac API

The TPac API provides a REST service for deploying tpacs in TM and downloading/deleting tpac archive files.

Security

To access the tpac REST endpoint the caller needs to be authenticated using HTTP Basic Authentication. The authenticated user will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission '`REST Transactions API`'.

If the user is not a global administrator, only tpacs belonging to the organizations assigned to the user will be accessible.

URL Endpoint

The REST TPac URL endpoint is as follows:

```
https://<TM server>/manager/secure/rest/tpac/v1/
```

`TPac API`

This section provides a description all the REST TPac API operations including:

- POST Import TPac Archive
- DELETE TPac Archive
- GET TPac Archive

POST Import TPac Archive

Import a TPac archive ZIP file into the TM server.

URL Format

```
POST https://<tm server>/manager/secure/rest/tpac/v1/<client code>/
```

POST Body

The Multipart POST request parameters (hedaers) include:

- `archiveFile` - the tpac archive ZIP file (required)
- `override` - boolean header that determines whether to override existing tpac. Values: true, false (default false)

Example

Please note POST example below is idealized to provide an illustration of multipart POST request.

```
POST /manager/secure/rest/tpac/v1/maguire/ HTTP/1.1
Host: https://transact.maguire.com
Content-type: multipart/form-data, boundary=AaB03x

--AaB03x
content-disposition: form-data; name="archiveFile"; filename="tpac-fis-chexsystems-v1-0-001.zip"
Content-Transfer-Encoding: binary
...
--AaB03x--
==========================
```

`200 Response`

```
{
  "archiveName": "tpac-fis-chexsystems-v1-0-001.zip",
  "importMessage": "Imported services for tpac 'FIS ChexSystems' for organization 'client' to Organization
'client' successfully.",
```

```
  "importStatus": "Completed",
  "importTime": "2017-05-19T16:32+1000"
}
```

DELETE TPac

Irrevocably delete an tpac. **USE THIS OPERATION WITH CAUTION**.

URL Format

```
https://<tm server>/manager/secure/rest/tpac/v1/<client code>/<tpac project name>/<tpac version>
```

DELETE Examples

```
DELETE https://transact.maguire.com/manager/secure/rest/tpac/v1/maguire/project-name/1.0
```

200 Response

The response will not contain any additional information. If the status code is 200, the tpac was deleted successfully.

GET TPac

Download a tpac.

URL Format

```
https://<tm server>/manager/secure/rest/tpac/v1/<client code>/<tpac project name>/<tpac version>
```

GET Examples

```
GET https://transact.maguire.com/manager/secure/rest/tpac/v1/maguire/project-name/1.0
```

200 Response

The response will return the tpac archive file.

Transactions API

REST Transactions API

The Transactions API provides a REST service for querying and updating transactions in Transaction Manager in a secure and efficient manner.

Security

To access the Transactions REST endpoint the caller needs to be authenticated using HTTP Basic Authentication. The authenticated user will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission '**`REST Transactions API`**'. You will also need to be either an administrator with global access or be associated with an organization to access its transaction records.

URL Endpoint

The REST Transactions API URL endpoint is as follows:

```
https://<TM server>/manager/secure/rest/transactions/v1/
```

Service API

This section provides a description all the REST Transactions API operations including:

- GET Transaction
- GET Transactions
- GET Portal Help Desk
- GET Portal User Submissions
- GET Portal User ToDo
- POST Create Task
- PUT Update Transaction

GET Transaction

Retrieve a single transaction by specifying the database ID.

URL Format

```
GET https://<tm server>/manager/secure/rest/transactions/v1/<submission ID>
```

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/transactions/v1/71139
```

200 Response

```
{
  "id": 71139,
  "trackingCode": "YG7M4B",
  "receiptNumber": "job-example-2-plus-5",
  "submitKey": "de838352020015c22826e1017b45fb83",
  "taskFlag": true,
  "taskKey": "de838352020015c22826e1017b45fb83",
  "taskType": "Review",
  "taskCreatedTimestamp": "2015-09-23T16:09:35+10:00",
  "taskSubject": "Additional review of Job Example 2 Plus from Anne Applicant",
  "taskMessage": "The application has been initially approved and is pending your final review.",
  "job": {
    "id": 1190,
    "jobName": "Job Controller - 2 Step Review",
    "referenceNumber": "QKQP4K",
    "jobKey": "2b76a98e70fa911f0c46ea73c624ce6c",
    "jobStep": "Additional Review",
    "jobAction": "Create Task"
  },
```

```
  "formCode": "job-example-2-plus",
  "formName": "Job Example 2 Plus",
  "formVersion": "1.0",
  "clientCode": "maguire",
  "clientName": "Maguire",
  "portalName": "Maguire",
  "fieldWorkerFlag": false,
  "testMode": false,
  "userLoginName": "mike",
  "submissionGroupNames": "Job Managers",
  "formStatus": "Completed",
  "timeRequest": "2015-09-23T16:09:45+10:00",
  "timeSubmission": "2015-09-23T16:09:53+10:00",
  "timeLastUserActivity": "2015-09-23T16:09:53+10:00",
  "timeFormCompleted": "2015-09-23T16:09:53+10:00",
  "timeToSubmitSec": 7,
  "ipAddress": "123.134.145.156",
  "requestCookie": "JSESSIONID=l6Ijn4jARdI59Mz89mwgOJiq; __utma=264897119.2043164291...",
  "referer": "https://transact.maguire.com/maguire/secure/account/todo.htm",
  "userAgent": "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:40.0) Gecko/20100101 Firefox/40.0",
  "userAgentBrowser": "Firefox 40",
  "userAgentBrowserType": "Firefox",
  "userAgentBrowserVersion": "40",
  "userAgentDeviceType": "Desktop",
  "userAgentOs": "Windows 10",
  "userAgentOsType": "Windows",
  "userAgentOsVersion": "10",
  "dataExtracts": {
    "First Name": "Anne",
    "Last Name": "Applicant",
    "Email": "test@email.com",
    "Loan Amount": "123456",
    "routeName": "Approve"
  },
  "deliveryStatus": "Not Ready",
  "dataDeleted": false
}
```


GET Transactions

Retrieve the list of transactions sorted by database ID in descending order (the ordering can be customized).

URL Format

```
GET https://<tm server>/manager/secure/rest/transactions/v1/
```

Request Parameters

The table below shows the set of request parameters can be used as additional search criteria.
If multiple parameters are specified, all of them are applied to the result set.

Note: The special parameter value "null" can be used to specify that transactions are included only if the corresponding field is not set to a value.
For example, passing in the parameter "paymentStatus=null" would exclude all transactions for which payment has been requested, because those submissions would have a specific value for the payment status.

Parameter	Description
abandonmentFormStatus	filters on the form status of an abandoned transaction at abandonment time (Values: null, Assigned, Opened, Saved, Submitted)
attachmentsStatus	filters on the attachment status of the submission (Values: null, Required, Optional, Completed)
clientCode	filters on the organization code
clientName	filters on the organization name
contactEmailAddress	filters on the contact email address extracted from transaction data
contactPhone	filters on the contact phone number extracted from transaction data
deletedFlag	a boolean parameter that filters on the transaction data deleted flag (Values: true, false)
deliveryStatus	filters on the delivery status of the submission (Values: null, Not Required, Not Ready, Ready, Sent Email, Sent WS, In Progress, Pending, Completed, Error, Undeliverable)
endDate	only transactions requested before the specified end date will be included (Format: yyyy-MM-dd)
externalProfileId	filters on the external profile ID of the submission (only relevant if set by an external system or Groovy script)

fetchLimit	the maximum number of transactions to return (range 1 - 10000). Will default to 1000 if a valid value was not provided.
fieldWorkerFlag	a boolean parameter that filters on the set of spaces associated with a submission (Values: true, false); if true, the set of spaces must contain a TransactField space
formCode	filters on the form code
formName	filters on the form name
formStatus	filters on the form status of the submission (Values: null, Assigned, Opened, Saved, Submitted, Completed, Expired, Abandoned) Note that you can specify multiple values like this: [Saved, Submitted]
formVersion	filters on the form version number (e.g. "1.0") of the transaction
id	filters on the transaction database ID
jobKey	filters on the job key (which is set only if the transaction is associated with a collaboration job) Note: If you would like to exclude transactions associated with collaboration jobs, you can set this parameter to ""
loginName	filters on the login name of the user associated with the transaction (if set to "null", only transactions not associated with a user will be returned)
orderByAsc	specifies a field to use for result sorting in ascending order Note: Sorting will be case sensitive for string fields.
orderByDsc	specifies a field to use for result sorting in descending order (takes effect only if the parameter "orderByAsc" was not set) Note: Sorting will be case sensitive for string fields.
paymentStatus	filters on the payment status of the transaction (Values: null, Required, Pending, Completed, Error)
portalName	filters on the name of a space associated with the transaction
receiptNumber	filters on the transaction receipt number
receiptStatus	filters on the receipt status of the transaction (Values: null, Completed, Error, Error - No Data)
referenceNumber	filters on the job reference number (which is set only if the transaction is associated with a collaboration job)
startDate	only transactions requested on or after the specified start date will be included (Format: yyyy-MM-dd)
submitKey	filters on the transaction submit key (a GUID)
taskFlag	a boolean parameter that filters on whether the transaction is a task (Values: true, false)
taskKey	filters on the transaction task key (a GUID)
testMode	a boolean parameter that filters on the transaction test mode flag (Values: true, false)
trackingCode	filters on the transaction tracking code
transRefNumber	filters on the transaction reference number

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/transactions/v1/?startDate=2015-09-28&formCode=credit-card
```

200 Response


```
[
  {
    "id": 71865,
    "trackingCode": "B25LWD",
    "receiptNumber": "attachments-for-diff-1",
    "submitKey": "7b44754c1c8e1fc4ecd9dd0d0ecd5319",
    "taskFlag": false,
    "formCode": "credit-card",
    "formName": "Platinum Credit Card",
    "formVersion": "1.0",
    "clientCode": "maguire",
    "clientName": "Maguire Finance",
    "portalName": "Web Plugin",
    "fieldWorkerFlag": false,
    "testMode": false,
    "formStatus": "Completed",
    "timeRequest": "2015-09-28T16:13:53+10:00",
    "timeSubmission": "2015-09-28T16:14:00+10:00",
    "timeLastUserActivity": "2015-09-28T16:14:00+10:00",
    "timeFormCompleted": "2015-09-28T16:14:00+10:00",
    "timeToSubmitSec": 7,
    "ipAddress": "123.134.145.156",
    "requestCookie": "JSESSIONID=HD3Tet57kC-Pqda7+jGO7ubT; _ga=GA1.2.1363876154.1441243068;",
    "referer": "https://transact.maguire.com/manager/admin/form/form-edit.htm?entityId=5013",
    "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)...",
    "userAgentBrowser": "Edge 12",
    "userAgentBrowserType": "Edge",
    "userAgentBrowserVersion": "12",
    "userAgentDeviceType": "Desktop",
    "userAgentOs": "Windows 10",
```

```
    "userAgentOsType": "Windows",
    "userAgentOsVersion": "10",
    "deliveryStatus": "Ready",
    "dataDeleted": false
  }
]
```

GET Portal Help Desk

Retrieve the list of help desk transactions similar to what the help desk page on a TM user space displays.

URL Format

```
GET https://<tm server>/manager/secure/rest/transactions/v1/portal/help-desk
```

Request Parameters

The table below shows the set of request parameters can be used as additional search criteria.
If multiple parameters are specified, all of them are applied to the result set.

Parameter	Description
formStatus	filters on the form status of the submission (Values: Assigned, Opened, Saved, Submitted, Completed, Cancelled, Abandoned)
includeAnonymousSubmissions	a boolean parameter that controls the inclusion of anonymous transactions (Values: true, false). By default, anonymous transactions will be included.
includeAuthenticatedSubmissions	a boolean parameter that controls the inclusion of authenticated transactions (Values: true, false). By default, authenticated transactions will be included.
keyword	only transactions fitting the keyword will be included
portalName	filters on the name of the space for which help desk submissions are to be retrieved Note: When invoking the service mounted on the management console (context path /manager), this parameter is required.
startDate	the search start date (Format: yyyy-MM-dd)

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/transactions/v1/portal/help-desk?
portalName=Maguire&formStatus=Submitted
```

200 Response

```
[
  {
    "id": 69033,
    "trackingCode": "HX4BG7",
    "receiptNumber": "smartform-test-1",
    "submitKey": "0cedffee9c9af536c24315fc06a9330b",
    "taskFlag": false,
    "formCode": "smartform-test",
    "formName": "SmartForm Test",
    "formVersion": "1.0",
    "clientCode": "MACQUARIE",
    "clientName": "Macquarie",
    "portalName": "Maguire",
    "fieldWorkerFlag": false,
    "testMode": false,
    "formStatus": "Submitted",
    "timeRequest": "2015-09-07T10:31:02+10:00",
    "timeSubmission": "2015-09-07T10:31:23+10:00",
    "timeLastUserActivity": "2015-09-07T10:31:24+10:00",
    "timeToSubmitSec": 22,
    "ipAddress": "123.134.145.156",
    "referer": "https://transact.maguire.com/maguire/landing.htm?formCode=smartform-test",
    "userAgent": "AcroForms",
    "userAgentBrowser": "AcroForms Unknown",
    "userAgentBrowserType": "AcroForms",
    "userAgentBrowserVersion": "Unknown",
    "userAgentDeviceType": "Unknown",
```

```
    "userAgentOs": "Unknown ",
    "userAgentOsType": "Unknown",
    "userAgentOsVersion": "Unknown",
    "paymentStatus": "Required",
    "paymentType": "Credit/Debit Card",
    "paymentTotal": 49.0,
    "deliveryStatus": "Not Ready",
    "dataDeleted": false
  },
  {
    "id": 69031,
    "trackingCode": "6QZ6JD",
    "receiptNumber": "smartform-pdf-test-50",
    "submitKey": "bc47ba43993152ec2503167ce117c75a",
    "taskFlag": false,
    "formCode": "smartform-pdf-test",
    "formName": "SmartForm PDF Test",
    "formVersion": "1.0",
    "clientCode": "maguire",
    "clientName": "Maguire",
    "portalName": "Maguire",
    "fieldWorkerFlag": false,
    "testMode": false,
    "formStatus": "Submitted",
    "timeRequest": "2015-09-07T10:29:11+10:00",
    "timeSubmission": "2015-09-07T10:29:29+10:00",
    "timeLastUserActivity": "2015-09-07T10:29:29+10:00",
    "timeToSubmitSec": 18,
    "ipAddress": "123.134.145.156",
    "referer": "https://transact.maguire.com/manager/admin/form/form-search.htm?_wid=d51",
    "userAgent": "AcroForms",
    "userAgentBrowser": "AcroForms Unknown",
    "userAgentBrowserType": "AcroForms",
    "userAgentBrowserVersion": "Unknown",
    "userAgentDeviceType": "Unknown",
    "userAgentOs": "Unknown ",
    "userAgentOsType": "Unknown",
    "userAgentOsVersion": "Unknown",
    "attachmentsStatus": "Required",
    "paymentStatus": "Required",
    "paymentType": "Credit/Debit Card",
    "paymentTotal": 39.95,
    "dataExtracts": {
      "Given Name": "Test",
      "Family Name": "User",
      "Email": "test@avoka.com"
    },
    "deliveryStatus": "Not Ready",
    "dataDeleted": false
  },
  {
    "id": 69030,
    "trackingCode": "Z4QJWA",
    "receiptNumber": "smartform-pdf-test-49",
    "submitKey": "8fc87249c79ca45cb083879b678133b1",
    "taskFlag": false,
    "formCode": "smartform-pdf-test",
    "formName": "SmartForm PDF Test",
    "formVersion": "1.0",
    "clientCode": "maguire",
    "clientName": "Maguire",
    "portalName": "Maguire",
    "fieldWorkerFlag": false,
    "testMode": false,
    "formStatus": "Submitted",
    "timeRequest": "2015-09-07T10:27:59+10:00",
    "timeSubmission": "2015-09-07T10:28:48+10:00",
    "timeLastUserActivity": "2015-09-07T10:28:48+10:00",
    "timeToSubmitSec": 79,
    "ipAddress": "123.134.145.156",
    "referer": "https://transact.maguire.com/manager/admin/form/form-search.htm?_wid=d51",
    "userAgent": "AcroForms",
    "userAgentBrowser": "AcroForms Unknown",
    "userAgentBrowserType": "AcroForms",
    "userAgentBrowserVersion": "Unknown",
    "userAgentDeviceType": "Unknown",
    "userAgentOs": "Unknown ",
    "userAgentOsType": "Unknown",
```

```
    "userAgentOsVersion": "Unknown",
    "attachmentsStatus": "Required",
    "paymentStatus": "Required",
    "paymentType": "Credit/Debit Card",
    "paymentTotal": 39.95,
    "dataExtracts": {
      "Given Name": "First",
      "Family Name": "Last"
    },
    "deliveryStatus": "Not Ready",
    "dataDeleted": false
  }
]
```

GET Portal User Submissions

Retrieve the list of submissions completed by a user similar to what the submissions page on a TM user space displays.

URL Format

```
GET https://<tm server>/manager/secure/rest/transactions/v1/portal/user-submissions
```

Request Parameters

The table below shows the set of request parameters can be used as additional search criteria.
If multiple parameters are specified, all of them are applied to the result set.

Parameter	Description
clientName	the name of the organization for which transactions shall be retrieved
endDate	the search end date (Format: yyyy-MM-dd)
formNameLike	filters on a substring of the form name
includeGroupItems	a boolean parameter that controls the inclusion of transactions associated with groups the user is a member of (Values: true, false). By default, anonymous transactions will be included.
keyword	only transactions fitting the keyword will be included
portalName	filters on the name of the space for which submissions are to be retrieved Note: When invoking the service mounted on the management console (context path /manager), this parameter is required.
startDate	the search start date (Format: yyyy-MM-dd)
userLoginName	filters on the user login name for which transactions are to be retrieved Note: This parameter is required.

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/transactions/v1/portal/user-submissions?
portalName=Maguire&userLoginName=someAdmin&formNameLike=Payment
```

200 Response

```
[
  {
    "id": 69618,
    "trackingCode": "AFG8TW",
    "receiptNumber": "pf-lt41-15",
    "submitKey": "3888c972a167ca55e967cd764ab691bf",
    "taskFlag": false,
    "formCode": "pf-lt41",
    "formName": "Payment Form",
    "formVersion": "1.0",
    "clientCode": "lt41",
    "clientName": "Test Client (4.1)",
    "portalName": "Maguire",
    "fieldWorkerFlag": false,
    "testMode": false,
    "userLoginName": "someadmin",
    "formStatus": "Completed",
```

```
    "timeRequest": "2015-09-11T09:40:42+10:00",
    "timeSubmission": "2015-09-11T09:40:45+10:00",
    "timeLastUserActivity": "2015-09-22T15:48:35+10:00",
    "timeFormCompleted": "2015-09-22T15:48:35+10:00",
    "timeToSubmitSec": 4,
    "ipAddress": "123.134.145.156",
    "requestCookie": "JSESSIONID=ZlSTyEAIR-LB0wGwCNIL1vk5; __utma=264897119.59525688.1351658449...",
    "referer": "https://transact.maguire.com/maguire/login.htm",
    "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.0",
    "userAgentBrowser": "Firefox 40",
    "userAgentBrowserType": "Firefox",
    "userAgentBrowserVersion": "40",
    "userAgentDeviceType": "Desktop",
    "userAgentOs": "Windows 7",
    "userAgentOsType": "Windows",
    "userAgentOsVersion": "7",
    "attachmentsStatus": "Completed",
    "deliveryStatus": "Sent Email",
    "deliveryMessage": "Sent Email Secure delivery notification to: test@avoka.com",
    "deliveryTime": "2015-09-22T15:48:50+10:00",
    "dataDeleted": false
  }
]
```


GET Portal User ToDo

Retrieve the list of to do items for a user similar to what the to do page on a TM user space displays.

URL Format

```
GET https://<tm server>/manager/secure/rest/transactions/v1/portal/user-todo
```

Request Parameters

The table below shows the set of request parameters can be used as additional search criteria.
If multiple parameters are specified, all of them are applied to the result set.

Parameter	Description
formStatus	filters on the form status of the submission (Values: Assigned, Opened, Saved, Submitted, Completed, Cancelled, Abandoned)
includeGroupItems	a boolean parameter that controls the inclusion of transactions associated with groups the user is a member of (Values: true, false). By default, anonymous transactions will be included.
keyword	only transactions fitting the keyword will be included
portalName	filters on the name of the space for which submissions are to be retrieved Note: When invoking the service mounted on the management console (context path /manager), this parameter is required.
userLoginName	filters on the user login name for which transactions are to be retrieved Note: This parameter is required.

URL Example

```
GET https://transact.maguire.com/manager/secure/rest/transactions/v1/portal/user-todo?
portalName=Maguire&userLoginName=someAdmin&formStatus=Saved
```

200 Response


```
[
  {
    "id": 63744,
    "trackingCode": "LFS2HH",
    "submitKey": "8113f5a32e0647321b8047c5c8034bb1",
    "taskFlag": true,
    "taskKey": "8113f5a32e0647321b8047c5c8034bb1",
    "taskType": "Form",
    "taskCreatedTimestamp": "2015-08-06T11:04:53+10:00",
    "taskSubject": "Test Task 4.0",
    "formCode": "pf-lt418",
    "formName": "Payment Form",
```

```
    "formVersion": "1.0",
    "clientCode": "lt418",
    "clientName": "Test Client (4.1.8)",
    "fieldWorkerFlag": false,
    "testMode": false,
    "userLoginName": "someadmin",
    "formStatus": "Assigned",
    "userAgentBrowser": "",
    "userAgentOs": "",
    "dataDeleted": false
  },
  {
    "id": 54315,
    "trackingCode": "2QJHMB",
    "receiptNumber": "pf-lt41-14",
    "submitKey": "b097f6a9878a27c5ad29844e97c7d237",
    "taskFlag": false,
    "formCode": "pf-lt41",
    "formName": "Payment Form",
    "formVersion": "1.0",
    "clientCode": "lt41",
    "clientName": "Test Client (4.1)",
    "portalName": "Maguire",
    "fieldWorkerFlag": false,
    "testMode": false,
    "userLoginName": "someadmin",
    "formStatus": "Submitted",
    "timeRequest": "2015-05-19T11:30:36+10:00",
    "timeSubmission": "2015-05-19T11:32:33+10:00",
    "timeLastUserActivity": "2015-05-19T11:32:55+10:00",
    "timeToSubmitSec": 117,
    "ipAddress": "114.141.100.203",
    "requestCookie": "JSESSIONID=iIsfRT+sa4I1T2inf0nk5M57; __utma=264897119.59525688.1351658449...",
    "referer": "https://transact.maguire.com/maguire/landing.htm?formCode=pf-lt41",
    "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0",
    "userAgentBrowser": "Firefox 37",
    "userAgentBrowserType": "Firefox",
    "userAgentBrowserVersion": "37",
    "userAgentDeviceType": "Desktop",
    "userAgentOs": "Windows 7",
    "userAgentOsType": "Windows",
    "userAgentOsVersion": "7",
    "paymentStatus": "Required",
    "paymentType": "Hosted",
    "paymentTotal": 112.0,
    "paymentLog": {
      "paymentStatus": "Cancelled",
      "paymentLogKey": "cab053fa940afb12fabce66a36ac0496",
      "paymentServiceCode": "PayPal",
      "userIpAddress": "114.141.100.203",
      "doTimestamp": "2015-05-19T11:32:41+10:00",
      "doAmount": 11200,
      "doMerchTxnRef": "9fb84dad611a5",
      "doVersion": "76.0",
      "drTxnResponseCode": "Cancelled",
      "drTxnResponseMsg": "The transaction has been cancelled by the user"
    },
    "deliveryStatus": "Not Ready",
    "dataDeleted": false
  }
]
```


POST Create Task

Create a task that can be assigned to a user or a group.

URL Format

```
POST https://<tm server>/manager/secure/rest/transactions/v1/
```

POST Body

The POST request must contain a request parameter named "task", whose value contains a JSON structure describing the task. To support multiple parameters this API uses the HTTP Content Type: **`application/x-www-form-urlencoded`**

The set of task JSON attributes is listed below (all attributes are optional unless indicated otherwise):

Attribute	Type	Description
allowClaimFlag	boolean	a flag describing whether the task will be claimable (relevant for tasks assigned to groups)
address	String	a string containing the physical address related to the task
contactEmailAddress	String	the email address to send email to (must be specified if sendEmailFlag is set to true)
datetimeExpiry	String	a JSON ISO DateTime string without milliseconds describing when the task should expire automatically if it has not been completed the required date format is yyyy-MM-dd'T'HH:mm:ssZZ
datetimeScheduled	String	a JSON ISO DateTime string without milliseconds describing when the task is scheduled the required date format is yyyy-MM-dd'T'HH:mm:ssZZ
emailMessage	String	the email message to use in the task notification email (used only if sendEmailFlag is set to true)
emailSubject	String	the email subject to use in the task notification email (used only if sendEmailFlag is set to true)
formCode	String	the form code of the form that the task will be based on (required)
formPrefillServiceName	String	the name of a service definition of type 'Form Prefill' that will be used to provide prefill XML data at task creation time (used only if taskType is "Anonymous")
groups	List of String	A list of group names to which the task shall be assigned (required if userLoginName is not set and taskType is not "Anonymous")
latitude	Double	The latitude of the location the task relates to
longitude	Double	The longitude of the location the task relates to
portalName	String	The name of the space that will be hosting the task (required if taskType is set to "Anonymous")
receiptNumber	String	The receipt number that will be used for the task transaction
reviewSubmissionId	String	The database ID of the submission that this task will review (required and used only if taskType is set to "Review")
saveChallengeAnswer	String	If you would like to secure the task further, you can specify a secret answer the user will have to enter to access the form (used only if taskType is set to "Anonymous")
sendEmailFlag	boolean	a flag controlling whether an email shall be sent to notify of the new task (if taskType is set to "Anonymous", the email will be sent to the contact email address)
sequence	Integer	The task sequence number (mainly used in collaboration job task sequencing)
taskMessage	String	The task message that will be displayed to assignees
taskSubject	String	The task subject that will be displayed to assignees
taskType	String	The type of task you want to create (required). Possible values: • Anonymous: Create an anonymous saved form • Form: Create a standalone task assigned to a user or one or more groups • Review: Create a task relating to a previous submission and assign it to a user or one or more groups
transRefNumber	String	A custom reference number that will be stored in the transaction object (used only if the taskType is set to "Anonymous")
userDeletableFlag	boolean	a flag controlling whether the task can be deleted by assignees
userLoginName	String	The login name of the assignee (used only if taskType is not "Anonymous")

In addition to the **`task`** request parameter, there are two optional parameters you can use to pass in XML task data:

- **`formDataXml`** - the XML data to be used as the seed file
- **`inputXmlData`** - the XML data that will be mapped into the form using XML prefill mappings set up in TM

The two XML parameters can be passed in as request parameters, or alternatively can be included in a Multipart POST request.

Example: Creating an anonymous saved form

This example creates an anonymous saved form secured with a secret code. The person who will fill in the form is notified via email. The example the message body below formatted for easier reading.

Please ensure you specify **`Content-Type: application/x-www-form-urlencoded`**

```
POST <b>/manager/secure/rest/transactions/v1/</b> HTTP/1.1
Host: https://transact.maguire.com
Content-Type: application/x-www-form-urlencoded

<b>task</b>={
    "taskType": "Anonymous",
    "portalName": "Maguire",
    "formCode": "pf-lt41",
    "taskSubject": "Please complete this submission",
    "taskMessage": "This submission has been assigned to you to complete.",
```

```
    "saveChallengeAnswer": "b3d5",
    "contactEmailAddress": "test@avoka.com",
    "sendEmailFlag": true,
    "formPrefillServiceName": "Task Prefill"
}
```

200 Response

```
{
  "id": 73003,
  "trackingCode": "7FAE76",
  "submitKey": "01b2b3c0ac09dc706fb81c0e47731724",
  "taskFlag": false,
  "taskCreatedTimestamp": "2015-10-06T14:04:40+11:00",
  "taskSubject": "Please complete this submission",
  "taskMessage": "This submission has been assigned to you to complete.",
  "formCode": "pf-lt41",
  "formName": "Payment Form",
  "formVersion": "1.0",
  "clientCode": "lt41",
  "clientName": "Test Client (4.1)",
  "portalName": "Maguire",
  "fieldWorkerFlag": false,
  "testMode": false,
  "contactEmailAddress": "test@avoka.com",
  "formStatus": "Saved",
  "timeSubmission": "2015-10-06T14:04:39+11:00",
  "userAgentBrowser": "",
  "userAgentOs": "",
  "deliveryStatus": "Not Ready",
  "dataDeleted": false,
  "formURL": "https://transact.maguire.com/maguire/servlet/SmartForm.html?
submitKey=01b2b3c0ac09dc706fb81c0e47731724"
}
```

Example: Creating a form task with XML form data

This example creates a standalone form task assigned to a single user with the **`task`** parameter, and includes the XML form data provided with the **`formDataXml`** parameter.

Please note the form prefill data is **`application/x-www-form-urlencoded`** encoded, and the message body below formatted for easier reading. In practice it may be eaiser to use a Multipart POST request to upload the form XML data.

```
POST <b>/manager/secure/rest/transactions/v1/</b> HTTP/1.1
Host: https://transact.maguire.com
Content-Type: application/x-www-form-urlencoded

<b>task</b>={
    "taskType": "Form",
    "formCode": "pf-lt41",
    "taskSubject": "Monthly activity summary",
    "taskMessage": "Please fill in the monthly activity summary for 2015-09",
    "transRefNumber": "ma-201509-jdoe",
    "sendEmailFlag": true,
    "emailSubject": "TASK: Monthly activity summary",
    "datetimeScheduled": "2015-10-09T09:00:00%2b10:00",
    "datetimeExpiry": "2015-10-16T23:59:59%2b10:00",
    "userLoginName": jdoe,
    "userDeletableFlag": false
}
&formDataXml=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22%3F%3E%0A%3CAvokaSmartForm%3E%0A++%
3CAddressAustralia%3E%0A++++%3CAddressLine1%2F...
```

200 Response

```
{
  "id": 73004,
  "trackingCode": "3ZYAA3",
  "submitKey": "d97d8a4be640d6a04cec039a1a657971",
  "taskFlag": true,
  "taskKey": "d97d8a4be640d6a04cec039a1a657971",
```

```
  "taskType": "Form",
  "taskCreatedTimestamp": "2015-10-06T14:09:31+11:00",
  "taskScheduledTimestamp": "2015-10-09T10:00:00+11:00",
  "taskExpiryTimestamp": "2015-10-17T00:59:59+11:00",
  "taskSubject": "Monthly activity summary",
  "taskMessage": "Please fill in the monthly activity summary for 2015-09",
  "formCode": "pf-lt41",
  "formName": "Payment Form",
  "formVersion": "1.0",
  "clientCode": "lt41",
  "clientName": "Test Client (4.1)",
  "fieldWorkerFlag": false,
  "testMode": false,
  "userLoginName": "jdoe",
  "formStatus": "Assigned",
  "userAgentBrowser": "",
  "userAgentOs": "",
  "dataDeleted": false
}
```

Example: Creating a review task

This example creates a review task assigned to a group of users.

```
POST <b>/manager/secure/rest/transactions/v1/</b> HTTP/1.1
Host: https://transact.maguire.com
Content-Type: application/x-www-form-urlencoded

<b>task</b>={
    "taskType": "Review",
    "formCode": "pf-lt41",
    "taskSubject": "Please review application no. 69618",
    "sendEmailFlag": false,
    "address": "The Corso, Manly NSW 2095",
    "latitude": -33.86,
    "longitude": 151.2,
    "userDeletableFlag": false,
    "allowClaimFlag": true,
    "groups": [
        "Job Reviewers",
        "Job Managers"
    ],
    "reviewSubmissionId": "69618"
}
```

200 Response

```
{
  "id": 73008,
  "trackingCode": "KVGGBB",
  "submitKey": "13537f9ab228a8bc50eae0e5d8bb3436",
  "taskFlag": true,
  "taskKey": "13537f9ab228a8bc50eae0e5d8bb3436",
  "taskType": "Review",
  "taskCreatedTimestamp": "2015-10-06T14:14:49+11:00",
  "taskSubject": "Please review application no. 69618",
  "taskAddress": "The Corso, Manly NSW 2095",
  "taskLatitude": -33.86,
  "taskLongitude": 151.2,
  "formCode": "pf-lt41",
  "formName": "Payment Form",
  "formVersion": "1.0",
  "clientCode": "lt41",
  "clientName": "Test Client (4.1)",
  "fieldWorkerFlag": false,
  "testMode": false,
  "submissionGroupNames": "Job Managers, Job Reviewers",
  "formStatus": "Assigned",
  "userAgentBrowser": "",
  "userAgentOs": "",
  "attachmentsStatus": "Optional",
  "dataDeleted": false
}
```

PUT Update Transaction

Update the status of a transaction.

URL Format

```
PUT https://<tm server>/manager/secure/rest/transactions/v1/<transaction id>
```

PUT Body

The body of your PUT request must contain a structure that looks like this:

```
{
    "formStatus": <new form status value>,
    "processingStatus": <new processing status value>
}
```

Both status values are optional.
The new form status can be "Saved" or "Abandoned".
The processing status value is added as a new processing status entry associated with the transaction (visible to the user), but does not otherwise influence transaction processing.

URL Example

```
PUT <b>https://transact.maguire.com/manager/secure/rest/transactions/v1/417</b>
```

```
{
    "formStatus": "Saved",
    "processingStatus": "Reverted to saved status at user request"
}
```

200 Response


```
{
  "id": 72492,
  "trackingCode": "P74L3V",
  "receiptNumber": "pf-lt41-16",
  "submitKey": "9f746b51721051b674b057c37b4bee95",
  "taskFlag": false,
  "formCode": "pf-lt41",
  "formName": "Payment Form",
  "formVersion": "1.0",
  "clientCode": "lt41",
  "clientName": "Test Client (4.1)",
  "portalName": "Maguire",
  "fieldWorkerFlag": false,
  "testMode": false,
  "userLoginName": "someadmin",
  "formStatus": "Saved",
  "timeRequest": "2015-10-02T14:07:39+10:00",
  "timeSubmission": "2015-10-02T14:10:25+10:00",
  "timeLastUserActivity": "2015-10-02T14:11:14+10:00",
  "timeToSubmitSec": 166,
  "ipAddress": "123.134.145.156",
  "requestCookie": "JSESSIONID=pfD+pEOezdTwyev-jINZSn1y; __utma=264897119.59525688.1351658449.1393997213...",
  "referer": "https://transact.maguire.com/maguire/login.htm",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.0",
  "userAgentBrowser": "Firefox 40",
  "userAgentBrowserType": "Firefox",
  "userAgentBrowserVersion": "40",
  "userAgentDeviceType": "Desktop",
  "userAgentOs": "Windows 7",
  "userAgentOsType": "Windows",
```

```
  "userAgentOsVersion": "7",
  "attachmentsStatus": "Optional",
  "paymentStatus": "Required",
  "paymentType": "Hosted",
  "paymentTotal": 123.0,
  "processingStatus": "Reverted to saved status at user request",
  "processingStatusTime": "2015-10-02T14:11:14+10:00",
  "deliveryStatus": "Not Ready",
  "dataDeleted": false,
  "formURL": "https://transact.maguire.com/maguire/secure/servlet/SmartForm.html?
submitKey=9f746b51721051b674b057c37b4bee95"
}
```

Transaction History API

REST Transaction History API

The Transaction History API provides a REST service for querying the OLAP Transaction History table in a secure and efficient manner.

Security

To access the Transaction History REST endpoint the caller needs to be authenticated using HTTP Basic Authentication. The authenticated user will need an user account on the TM server and this account will need to be active and have access to the Management Console module.

To be authorized to call the service the user account will also need the Management Console permission '**`REST Transaction History API`**'. You will also need to be either an administrator with global access or be associated with an organization to access its transaction records.

URL Endpoint

The REST Transaction History API URL endpoint is as follows:

```
https://<TM server>/manager/secure/rest/transaction-history/v1/
```

Service API

This section provides a description all the REST Transaction History API operations including:

- GET Transaction History
- PUT Confirm Published

GET Transaction History

This API supports a single GET method, but provides a great deal of flexibility through a query builder API. Key features of this API include:

- filter criteria - to select individual transactions or return filtered transaction records
- output formats - to specify JSON or CSV output results
- fetch limits and offsets - enabling you to page through very large sets of transaction history records
- columns filter - to specify only the column values you want to see

URL Format

```
GET https://<tm server>/manager/secure/rest/transaction-history/v1/
```

Request Parameters

The table below shows the set of request parameters can be used as additional search criteria. If multiple parameters are specified, all of them are applied to the result set.

Parameter	Description
abandonmentFormStatus	filters on the abandonment form status of an abandoned transaction at abandonment time [Assigned, Opened, Saved, Submitted]
clientCode	filters on the organization code. Note this column is not indexed and using **`clientOid`** will be more efficient.
clientOid	filters on the organization client OID primary key
dwhPublishStatus	filters on data warehouse publish status, which can be set via the PUT operation This allows you to retrieve only values that you haven't previously retrieved and marked as published (via PUT) Possible values: null (to retrieve only unpublished values), Completed (to retrieve only previously published values)
endDate	only transactions requested before the specified **`time_request`** end date will be included (format: yyyy-MM-dd) For example the follow query parameters **`&startDate=2016-01-01&endDate=2016-06-30`** includes transaction between 1st January 2016 and 30th June 2016.
formCode	filters on the form code. Note this column is not indexed and using **`formOid`** will be more efficient.
formOid	filters on the form OID primary key
formStatus	filters on the form status of the submission [Assigned, Opened, Saved, Submitted, Completed, Expired, Abandoned]
formVersionNumber	filters on the form version number (e.g. "1.0") of the transaction
portalName	filters on the form space (portal) associated with the transaction
receiptNumber	filters on the transaction receipt number
referer	filters on the transaction referer using a like "%referer%" where clause

startDate	only transactions requested equal to or after the specified `time_request` end date will be included (format: yyyy-MM-dd). For example the follow query parameters `&startDate=2016-01-01&endDate=2016-06-30` includes transaction between 1st January 2016 and 30th June 2016.
submitKey	filters on the transaction submit key (GUID). Please note this column is not indexed.
submissionOid	filters on the transaction submission OID primary key. Please note this column is not indexed.
trackingCode	filters on the transaction tracking code, also known as the tracking number
trackingNumber	filters on the transaction tracking number, also known as the tracking code
userAgentDeviceType	filters on the User Agent Device Type [Desktop, Tablet, Mobile]
userAgentBrowserType	filters on the User Agent Browser Type
userAgentBrowserVersion	filters on the User Agent Browser Version
userAgentOsType	filters on the User Agent OS Type
userAgentOsVersion	filters on the User Agent OS Version
fetchLimit	the maximum number of transactions to return (range 1 - 100K). Will default to 1000 if a valid value was not provided. For example: `&fetchLimit=10000`
fetchOffset	the query fetch offset in rows. Use the fetch offset to page through very large result sets. For example: `&fetchLimit=10000&fetchOffset=20000` will return the next 10K, after the 20K record offset
format	specify the output format [JSON, CSV], if not specified a JSON array will be returned
columns	specify the columns to return in a comma separated format, e.g. `&columns=[form_code, form_status, time_request]`

GET Example 1

The GET example below uses the filter criteria:

- `formCode=FTX-CCA` - filter to include only transactions with the form code "FTX-CCA"
- `formStatus=Completed` - filter to include only transactions with the form status of "Completed"
- `fetchLimit=1` - filter to only return 1 records

```
GET https://transact.maguire.com/manager/secure/rest/transaction-history/v1/?formCode=FTX-
CCA&formStatus=Completed&fetchLimit=1
```

200 Response


```
[
  {
    "attachment_count": 0,
    "attachments_total_size": 0,
    "authenticated_flag": false,
    "client_code": "maguire",
    "client_name": "Maguire",
    "client_oid": 1,
    "content_length": "19163",
    "duration_client_response": 9,
    "duration_render": 253,
    "duration_total": 671,
    "far_data_version": "4.3.0sp2",
    "field_worker_flag": false,
    "form_code": "FTX-CCA",
    "form_name": "Credit Card Application",
    "form_oid": 2,
    "form_status": "Completed",
    "form_version_number": "3.0",
    "ip_address": "127.0.0.1",
    "pdf_submit_flag": false,
    "portal_name": "Maguire",
    "receipt_number": "FTX-CCA-2",
    "receipt_render_duration": 3497,
    "referer": "http://localhost:9080/manager/admin/form/form-search.htm?_wid=1101",
    "request_cookie": "JSESSIONID=Swc8R3ul6lbq8JvwCOTHajRD",
    "request_key": "618124c328c44eb376dd2f7a32e4081c",
    "request_log_count": 1,
    "request_url": "http://localhost:9080/maguire/servlet/SmartForm.html",
    "revision_number": 5,
    "submission_oid": 10,
    "submit_key": "1725c033d977b52bea9aef934cd63a58",
    "submitted_offline_flag": false,
    "task_flag": false,
    "time_form_completed": "2016-02-16T12:35+1100",
```

```
    "time_request": "2016-02-16T12:35+1100",
    "time_submission": "2016-02-16T12:35+1100",
    "time_to_submit_sec": 17,
    "tracking_number": "GD4ZFB",
    "transaction_history_oid": 4,
    "user_agent": "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:44.0) Gecko/20100101 Firefox/44.0",
    "user_agent_browser_type": "Firefox",
    "user_agent_browser_version": "44",
    "user_agent_device_type": "Desktop",
    "user_agent_os_type": "Windows",
    "user_agent_os_version": "10"
  }
]
```

GET Example 2

The GET example below uses the filter criteria:

- **`formCode=FTX-CCA`** - filter to include only transactions with the form code "FTX-CCA"
- **`fetchLimit=10`** - filter to only return 10 records
- **`format=CSV`** - output format CSV
- **`columns=[form_name,form_version_number,portal_name,submission_oid,tracking_number]`** - only include columns [form_name, form_version_number, portal_name, submission_oid, tracking_number]

```
GET https://transact.maguire.com/manager/secure/rest/transaction-history/v1/?formCode=FTX-
CCA&fetchLimit=10&format=CSV&columns=[form_name,form_version_number,portal_name,submission_oid,tracking_number]
```

200 Response

```
"form_name","form_version_number","portal_name","submission_oid","tracking_number"
"Credit Card Application","3.0","Maguire",10,"GD4ZFB"
"Credit Card Application","3.0","Maguire",13,"7T5QJF"
"Credit Card Application","3.0","Maguire",27,"RC93C9"
"Credit Card Application","3.0","Maguire",612,"Y3RZZL"
"Credit Card Application","3.0","Maguire",617,"GTS32H"
"Credit Card Application","3.0","Maguire",716,"GC2HNB"
"Credit Card Application","3.0","Maguire",719,"9P8W56"
"Credit Card Application","3.0","Maguire",720,"VGLHKW"
"Credit Card Application","3.0","Maguire",721,"TP2G7P"
"Credit Card Application","3.0","Maguire",722,"DTD65S"
```

PUT Confirm Published

Confirm multiple transaction history records as published to your data warehouse. Once records have been marked as published, they can be excluded from future GET requests using the **`dwhPublishStatus`** parameter.

PUT Request

```
PUT /manager/secure/rest/transaction-history/v1/ HTTP/1.1
Host: https://transact.maguire.com
Content-Type: application/json

{
    "transactionHistoryOid": [<transaction history oid 1>, <transaction history oid 2>, ...]
}
```

The number list of **`transactionHistoroyOid`** values must be specified and contain at least one value.
TM will find all transaction history entries with these OIDs and set their data warehouse publish status to "Completed" and the publish time to the current date time.

PUT Example

In this example we will get the list of transaction history entries that have not yet been marked as published (call 1), mark some of them as published (call 2), and again retrieve the now smaller set of unpublished transaction history entries (call 3).
Finally, we will get the list of previously published entries.

Call 1: GET request prior to PUT

```
GET https://transact.maguire.com/manager/secure/rest/transaction-history/v1/?dwhPublishFlag=null
```

Call 1: 200 Response

The result of this call has been heavily edited for brevity. For our purposes, it is enough to show which records are returned.

```
[
  {
    "receipt_number": "FTX-CCA-2",
    "submission_oid": 10,
    "transaction_history_oid": 1,
    ...
  },
  {
    "receipt_number": "COA-11",
    "submission_oid": 11,
    "transaction_history_oid": 2,
    ...
  },
  {
    "receipt_number": "NN1434-6",
    "submission_oid": 15,
    "transaction_history_oid": 3,
    ...
  },
  {
    "receipt_number": "NN1434-7",
    "submission_oid": 17,
    "transaction_history_oid": 4,
    ...
  }
]
```

Call 2: PUT to mark records 1 and 2 as published

```
PUT /manager/secure/rest/transaction-history/v1/ HTTP/1.1
Host: https://transact.maguire.com
Content-Type: application/json

{
    "transactionHistoryOid": [1, 2]
}
```

Call 2: 200 Response

```
{
  "result": "success"
}
```

Call 3: Same GET request after PUT

```
GET https://transact.maguire.com/manager/secure/rest/transaction-history/v1/?dwhPublishFlag=null
```

Call 3: 200 Response

The result of this call has been heavily edited for brevity. For our purposes, it is enough to show which records are returned.

Note how records that we marked as published in the PUT call are not returned because of the **`dwhPublishFlag=null`** URL parameter.
You can always get the full list of records by omitting the **`dwhPublishFlag`** URL parameter altogether.

```
[
  {
```

```
    "receipt_number": "NN1434-6",
    "submission_oid": 15,
    "transaction_history_oid": 3,
    ...
  },
  {
    "receipt_number": "NN1434-7",
    "submission_oid": 17,
    "transaction_history_oid": 4,
    ...
  }
]
```

Call 4: GET request for previously published records

```
GET https://transact.maguire.com/manager/secure/rest/transaction-history/v1/?dwhPublishFlag=Completed
```

Call 4: 200 Response

The result of this call has been heavily edited for brevity. For our purposes, it is enough to show which records are returned.

Note how only records that we marked as published in the PUT call are returned because of the `dwhPublishFlag=Completed` URL parameter. These records also have the publish status and date set.

```
[
  {
    "dwh_publish_status": "Completed",
    "dwh_publish_time": "2016-05-17T15:32+1000",
    "receipt_number": "FTX-CCA-2",
    "submission_oid": 10,
    "transaction_history_oid": 1,
    ...
  },
  {
    "dwh_publish_status": "Completed",
    "dwh_publish_time": "2016-05-17T15:32+1000",
    "receipt_number": "COA-11",
    "submission_oid": 11,
    "transaction_history_oid": 2,
    ...
  }
]
```

Transact Fluent SDK Download

To download the Transact SDK for Transact Manager 17.0.0 and later please see the Transact SDK Download page.

Dependencies

The Transact Fluent SDK has the following dependencies:

- Java 1.8 SDK (version used by Transact Manager)
- Apache Ant support (usually provided by the IDE)

Download Latest Version

You should download the Transact Fluent SDK version that matches the version of your target Transact Manager environment.

The Transact Fluent SDK consists of 2 packages:

1. transact-fluent-api-5.1.7.zip (25MB)
   Contains all required API library files and dependencies, as well as Javadoc and build assets
2. transact-project-template-5.1.7.zip (6KB)
   A pre-configured project framework with Ant build processes for getting started quickly.

These 2 packages should be downloaded and unzipped into the same parent directory on your development drive (e.g. your IDE workspace folder).

API changes for this latest release are noted in the Transact Manager 5.1.7 Release Notes relating to module **Fluent SDK**.

IDE Setup Instructions

You can find instructions for configuring your specific IDE here:

- Desktop IDE Development with The Transact Fluent API

Upgrading Projects

If you created a project in an earlier version of the SDK you may run through the following procedure to upgrade your project to the latest SDK version:

1. Ensure you first run through the IDE setup instructions for the latest version of the SDK and create the global library as described.
2. In your existing project, adjust the classpath dependencies to point to the new global library:
   a. In IntelliJ, go to the **Project Structure** dialog under Modules and select the Dependencies tab.
   b. In Eclipse, go to the **Project Properties** dialog under Java Build Path and select the Libraries tab.
3. Edit your **build.properties** file and ensure it contains the **transact.api.dir** pointing to your new Fluent API root folder, e.g.:

```
transact.api.dir=../transact-fluent-api-5.1.7
```

4. Copy the **scaffold.xml** build file from the project template into your existing project root directory and open it in the Ant build tab.
5. Run the **scaffold** ant task in the **scaffold.xml** build file.
6. Run the **regenerate-all-build-xml** task in the **scaffold.xml** build file.

Older Versions

Version 5.1.6

- transact-fluent-api-5.1.6.zip
- transact-project-template-5.1.6.zip
- Transact Manager 5.1.6 Release Notes

Version 5.1.5

- transact-fluent-api-5.1.5.zip
- transact-project-template-5.1.5.zip
- Transact Manager 5.1.5 Release Notes

Version 5.1.4

- transact-fluent-api-5.1.4.zip
- transact-project-template-5.1.4.zip
- Transact Manager 5.1.4 Release Notes

Version 5.1.3

- transact-fluent-api-5.1.3.zip

- transact-project-template-5.1.3.zip
- Transact Manager 5.1.3 Release Notes

Version 5.1.2

- transact-fluent-api-5.1.2.zip
- transact-project-template-5.1.2.zip
- Transact Manager 5.1.2 Release Notes

Version 5.1.1

- transact-fluent-api-5.1.1.zip
- transact-project-template-5.1.1.zip
- Transact Manager 5.1.1 Release Notes

Version 5.1.0

- transact-fluent-api-5.1.0.zip
- transact-project-template-5.1.0.zip
- Transact Manager 5.1.0 Release Notes

Version 5.0.5

- SDK 5.0.5 is compatible with Transact Manager 5.0.6, 5.0.7, 5.0.8, 5.0.9 and 5.0.10 environments
- transact-fluent-api-5.0.5.zip
- transact-project-template-5.0.5.zip
- Transact Manager 5.0.5 Release Notes
- Transact Manager 5.0.6 Release Notes
- Transact Manager 5.0.7 Release Notes
- Transact Manager 5.0.8 Release Notes
- Transact Manager 5.0.9 Release Notes
- Transact Manager 5.0.10 Release Notes

Version 5.0.4

- transact-fluent-api-5.0.4.zip
- transact-project-template-5.0.4.zip
- Transact Manager 5.0.4 Release Notes

Version 5.0.3

- SDK 5.0.4 is compatible with Transact Manager 5.0.3 environments
- Transact Manager 5.0.3 Release Notes

Version 5.0.2

- transact-fluent-api-5.0.2.zip
- transact-project-template-5.0.2.zip
- Transact Manager 5.0.2 Release Notes

Version 5.0.1

- transact-fluent-api-5.0.1.zip
- transact-project-template-5.0.1.zip
- Transact Manager 5.0.1 Release Notes

Version 5.0 (initial release):

- transact-fluent-api-5.0.zip
- transact-project-template-5.0.zip
- Transact Manager 5.0 Release Notes