Avoka Transact

Transact for Salesforce

Unknown macro: 'redirect'

Search this documentation

Contents

Recently Updated Pages

Transact for Salesforce
May 15, 2019 • updated by Anonymous • view change

How to manage user licensing for the Salesforce App
May 15, 2019 • updated by Anonymous • view change

How to configure Transaction Manager to support the Salesforce App
May 15, 2019 • updated by Anonymous • view change

Version 1.11
May 15, 2019 • updated by Anonymous • view change

Version 1.10
May 15, 2019 • updated by Anonymous • view change

Popular Topics

- kb-how-to-article
- kb-troubleshooting-article
- release-notes
- salesforce
- consumer-key
- consumer-secret
- forms
- groups
- security
- security-token

About Avoka Transact for Salesforce

Unknown macro: 'redirect'

Avoka Transact for Salesforce was created in response to recurring feedback we received from our customer base.

Feedback around the 'Customer Experiences' produced by the Avoka Transact platform has always been very positive and we've been fortunate enough to assist many businesses transform the way they transact with their customers online. However, a number of our clients were worried about the 'Staff Experience' and didn't want their Sales / Customer Service staff to have to login to a separate system in order to initiate these online transactions and track their progress to completion. Many of these customer facing staff spend their day inside a CRM environment, becoming very productive in this environment, and for a growing number of our clients this CRM is Salesforce.com.

So Avoka Transact for Salesforce is really about improving the 'Staff Experience', specifically the staff who spend the bulk of their time working in Salesforce and dealing directly with the customer.

We asked ourselves some questions:

- What if Sales staff could send out an Application Form to one of their contacts, directly from the Contact record, and have it pre-filled with information from Salesforce?
- How could we make it easy for Customer Service staff to assist customers complete their transactions without having to leave the Salesforce environment?
- Could we record completed form-based transactions into the CRM such that they get added to the historical timeline of the customer relationship?

The Avoka Transact for Salesforce App is designed to answer these questions.

Objectives of the App

The primary objectives of the solution are to:

1. Provide a Salesforce embedded UI that allows users to view and manage transactions that reside on the Avoka Transact platform without having to access the platform directly, removing the need for them to leave the Salesforce environment.
2. Provide a generic integration channel between Avoka Transact and Force.com to avoid custom integration work for each use case.
3. Provide a simple framework for integration that enables Salesforce consulting partners and Salesforce administrators with the ability to more easily configure and control the receipt of data from and release of data to the Avoka Transact platform.

Setup Process

Detailed setup instructions are provided in the following section of this knowledge base:

- Setup Instructions - Avoka Transact for Salesforce

Support Matrix

Component	Support
Salesforce	All editions of Salesforce are supported All Avoka APEX code is written for API version 33
Avoka Transaction Manager	See the Version Information Section

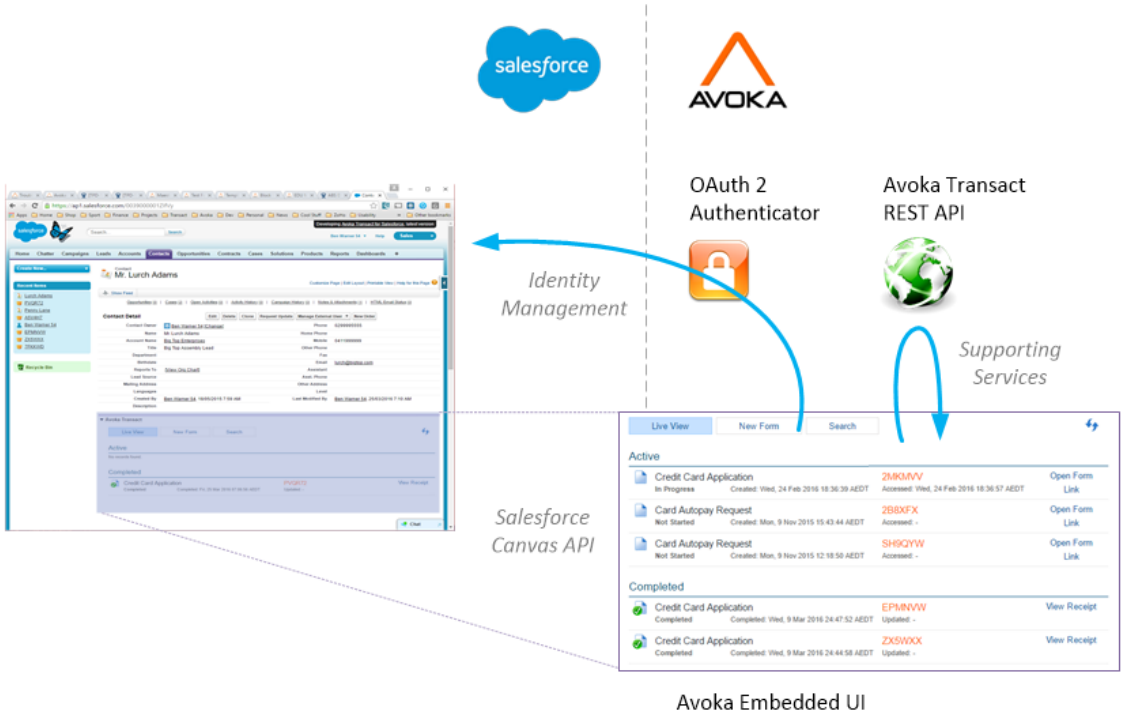
Administration Guidelines

How-to articles relating to administrative tasks can be found in the following section of this knowledge base:

- Administration Guide - Avoka Transact for Salesforce

User Interface Design

The user interface is facilitated by a Canvas application that is hosted on the Transaction Manager server and rendered into the Salesforce UI using the Canvas framework. A set of REST APIs exposed on the Avoka Transact Platform support the UI for accessing, creating and updating transactions.



Styling

Styling of the Avoka Transact Canvas UI is designed to blend in with the Salesforce Desktop UI by utilizing complementary colours, fonts and button styles. Where appropriate, subtle changes have been applied to provide a more modern look and feel, yet still maintaining a natural integration into the visual design of the platform.

Navigation

Page navigation is facilitated by a simple tab metaphor at the top of the UI, supporting navigation between the 4 available page functions:

1. **My Forms**: showing form transactions that are assigned to the current user. This page is typically only surfaced when the user accesses the UI via the Chatter feed.
2. **Live View**: showing form transactions that are linked to the current Salesforce object (SObject).
3. **New Form**: Used to select and create a new form transaction based on an available form template.
4. **Search**: Customer service style search facility to find forms using a search term. Results may include forms that are anonymous and unlinked.

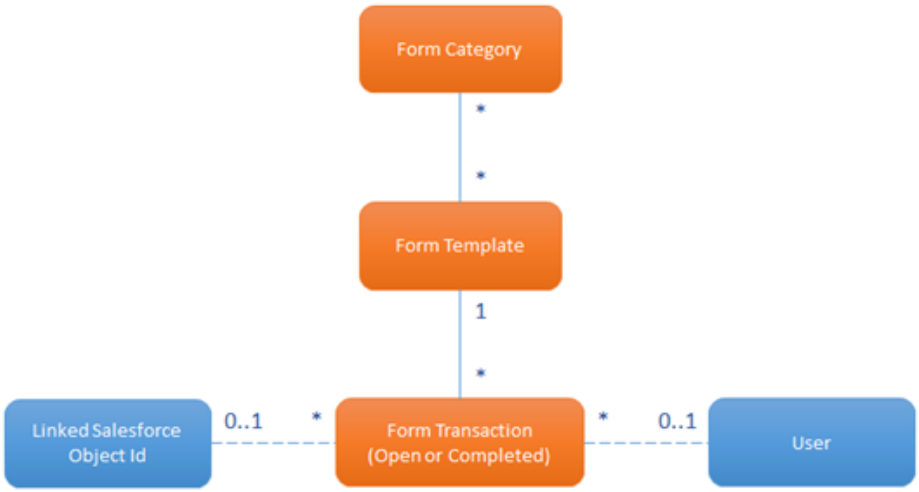
Information Architecture

A form template represents the code and configuration behind a form design and encapsulates the data model, business rules and presentation assets for a form. Within the Canvas UI, form templates are accessed by navigating to the ‘New Form’ page and selecting a form category. Form templates can be used to create new form transactions to be opened immediately or sent to a contact.

Form templates can belong to one or more form category. Form categories and the forms that belong to them are configured in Avoka Transaction Manager.

A form transaction may be assigned to a single user or assigned to no user (anonymous). Forms assigned to a user will appear under the ‘My Forms’ page when that user is logged in to Salesforce and accessing the Canvas UI.

A form transaction may be linked with a single Salesforce object or unlinked. Forms linked to a Salesforce object will appear under the ‘Live View’ page when viewing the detail layout for that object.



Note: Primary Entities in Avoka (orange) and Salesforce (blue) and their Relationships

Integration Approach

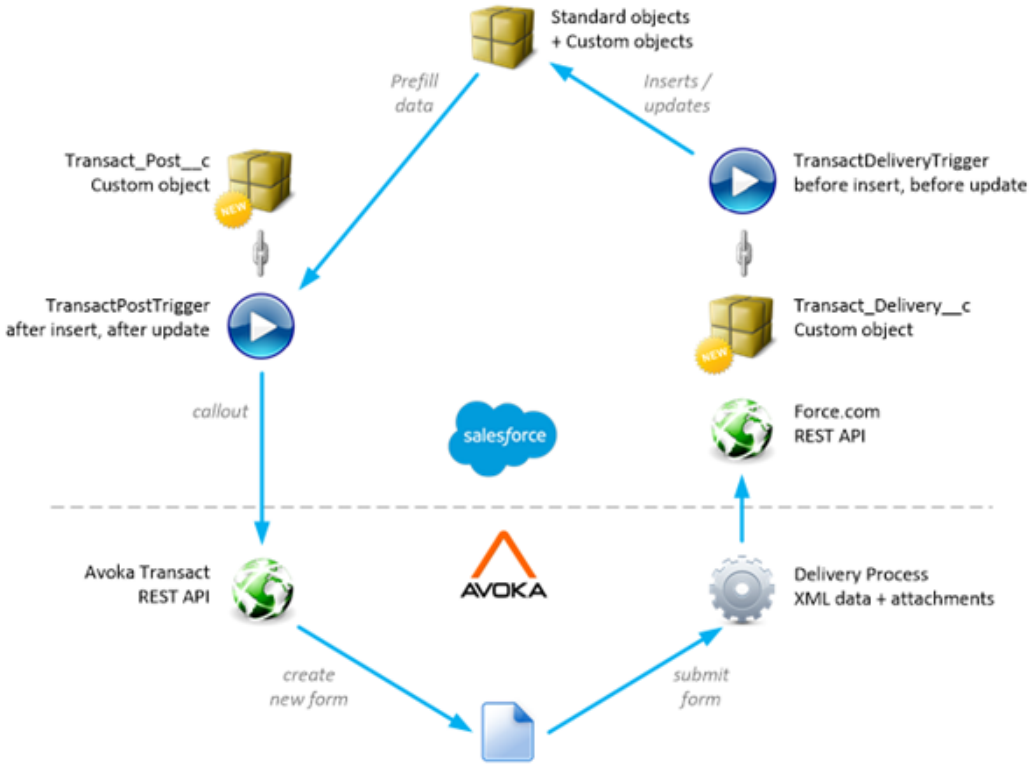
UI

The Avoka UI is built in and hosted on the Avoka Transact platform. This UI is integrated into the Salesforce page layouts by utilizing the Salesforce Canvas API.

APIs

API integration is facilitated by REST services on both platforms. Avoka Transact exposes REST services to accommodate the assignment of new forms from Salesforce, while REST APIs on the Salesforce side are utilized by Avoka Transact to deliver completed submissions.

Within Salesforce, these REST services are encapsulated by custom objects and associated triggers to facilitate the bi-directional communications, as illustrated in the following diagram.



Authentication / Identity Management

Authentication of Salesforce users into Avoka Transact is done via OAuth2. Users who access the Avoka UI must be authenticated with Salesforce. Upon first time access to the Avoka UI, a placeholder account is created on the Avoka Transact server with the required permissions to perform the functions associated with the UI.

Data Security

This section provides an assessment of the data security considerations associated with the managed package. For detailed information on data security in the Avoka Transact platform, request a copy of the Avoka Transact Security Architecture documentation.

The base installation of the package does not require access to any Salesforce standard or custom objects other than those contained within the package itself.

Control over which data is shared with Avoka Transact for prefill into forms, and which data is updated by incoming submissions is in complete control of the Salesforce administrators. This is a manual configuration function that is not enabled by default.

Data Access Controls

The solution provides a framework for Salesforce administrators to release data to Avoka Transact for the purposes of prefilling forms before sending them to contacts to complete and submit. This framework is facilitated by a set of Apex classes that consume configuration based rules to build dynamic SOQL queries for the retrieval of data.

Within this apex code, the following access controls are employed:

- **With Sharing**: All Apex classes in the package have the 'with sharing' control activated such that sharing rules of the current user are taken into account.
- **Data Access Rights**: Where configuration requires that a field be prefilled into a form, the current user's data access rights are first checked before that field is retrieved with SOQL. Similarly, where data flows back into Salesforce, the user's permission to create or update fields and objects is checked prior to performing any change. The Apex class 'SecurityController' has been included to accommodate these checks. The following code is an example of these checks in action:

```
if(SecurityController.isFieldAccessible(objectType, fieldName))
```

Off-Platform Data Storage

Avoka Transact is a System of Engagement, not a System of Record, so no data is retained in our platform (other than configuration data) longer than it is required in order to complete the data capture transaction and forward the data to the appropriate System of Record.

Due to the requirements of Avoka's most sensitive clients (Federal government agencies, police departments, financial services organisation), data security in Avoka Transact is exceptionally strong. All data in transit and at rest is encrypted according to the industry's highest standard and flexible purge policies allow clients to ensure that sensitive data is removed from the system at the earliest opportunity.

Salesforce Organization Impacts

Avoka Transact does not include pre-built data capture experiences targeting specific use cases, but rather provides a template based design environment with many pre-built components that can be strung together to rapidly produce these experiences. Each of Avoka's customers use Avoka Transact to facilitate different use cases within their organization, and these are typically built to spec.

With this consideration, it is difficult to provide a general assessment of the impacts the Salesforce package will have on the client's Salesforce Organization. Rather, the impacts need to be assessed on a case by case basis and Avoka will work with each client to ensure that the impacts understood.

The following sections outline the potential areas of impact and the considerations relating to each.

Licensing

The Avoka Transact for Salesforce package can be used on any edition of Salesforce.

Licenses are made available for a monthly fee on a per user basis. Any user who accesses any component included in the Managed Package must be licensed to use the solution. The primary functions required by users will be:

- Utilise the Canvas App to manage forms against a Salesforce object.
- Access delivery records resulting from Avoka submissions.
- Trigger functions within their Salesforce environment that in turn create a post request to Avoka Transact to assign a form.

Data Storage Requirements

The following data storage considerations need to be assessed on a case by case basis:

- Transaction Volumes: The volume of transactions being produced from the Avoka Transact platform and delivered into Salesforce will be a key factor in the determination of data storage requirements.
- Binary attachments: Where transactions call for binary attachments to be provided, and these attachments are designated for delivery into Salesforce, the storage requirements for these binary attachment needs to be considered. Controls can be put in place in Avoka Transact to restrict the size of attachments that may be provided by users.
- Purge Strategies: The data in the Avoka Transact custom object records is transient by design. Data purge strategies can be effectively applied to remove records that are no longer required. Purge rules may be developed to remove only the binary attachments and submission XML while maintaining the metadata about each delivered submission.

API Usage Volumes

The following API usage considerations need to be assessed on a case by case basis:

- Transaction Volumes: The volume of transactions being produced from the Avoka Transact platform and delivered into Salesforce will be a key factor in the determination of api usage requirements. Each submission delivered into Salesforce will utilize a minimum of 1 API call, but more where attachments are to be delivered also.
- Prefill Generation: Where a user is interacting with the Canvas App and attempts to send a form to a contact, this requires an API call to the prefill generation service.
- Where Transact Post records are created as a result of events in the Salesforce environment, these trigger callouts to the Avoka Transact platform.

Batch Processing

There are currently no batch processes included in the package. Clients may choose to implement their own purge process to remove post and delivery records older than a certain date.

Reporting

There are currently no reports or dashboards included in the package.

Setup Instructions - Avoka Transact for Salesforce

The how-to articles in this section provide instructions on setting up the Salesforce app to work with an instance of Avoka Transact and vise versa.

- How to configure Salesforce for integration with Avoka Transact
  - How to enable access to the Avoka Transact app via the App Menu
  - How to manage user licensing for the Salesforce App
  - How to retrieve the Consumer Key and Secret for a Salesforce Connected App
  - How to retrieve the Security Token for a Salesforce User Account
  - How to remove password expiry on a Salesforce User Account
- How to configure the Transact UI in Salesforce
- How to configure Transaction Manager to support the Salesforce App

How to configure Salesforce for integration with Avoka Transact

Unknown macro: 'redirect'

This article describes the process for configuring SalesForce when integrating with Avoka Transact for the first time.

Summary

The process for configuring your Salesforce environment for Avoka Transact API integration can be summarized as follows:

1. Install the Avoka Transact for Salesforce app
2. Create the Avoka Transact API User Account in Salesforce
3. Create the Avoka Transact Connected App in Salesforce
4. Provide the User Account and Connected App details to the Avoka Transact Team

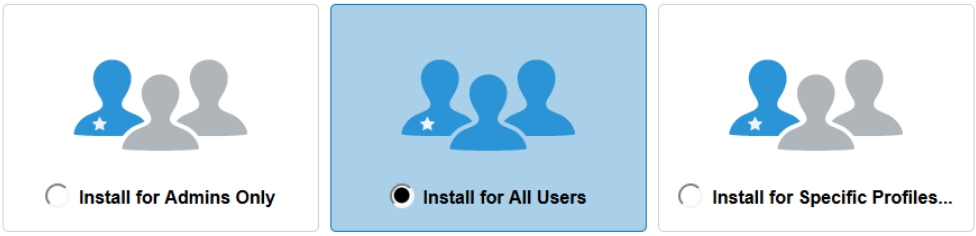
Install the Avoka Transact for Salesforce app

Install the latest version of the Avoka Transact for Salesforce app that supports your version of Transaction Manager:

- Versions & Installation Assets

To commence installation of a Salesforce package, upon opening the package URL in the browser you are required to login to the Salesforce environment that you wish to install into.

When installing the package it is best to elect to Install for All Users as shown below. This will avoid users being unable to access the app due to insufficient privileges.



If you have already installed the app for Admins Only or Specific Profiles and some users are unable to access the app due to 'Insufficient Privileges', you can resolve this issue as described in this troubleshooting article:

- Insufficient Privileges: You do not have the level of access necessary to perform the operation you requested...

Creating the Avoka Transact API User Account in Salesforce

We recommend you create a separate Salesforce user account dedicated to Avoka Transact integrations. This allows you to keep your real user credentials private and lets you restrict the permissions of the integration user as appropriate.

To create the user and its permission profile for API level integration, please contact your Salesforce administrator or refer to the Salesforce documentation. The following items should be considered when doing so:

1. Suggested User Account attributes:

Attribute	Value
First Name	*Blank*
Last Name	**Avoka Transact**
Alias	**transact**
Email	*Suggest using the email address of your Salesforce administrator*
Username	**transact@yourdomain.com.<environment>** *Append the environment name for non-prod environments*
Nickname	**Avoka Transact**

2. The user must have API access enabled
   See also: https://help.salesforce.com/HTViewHelpDoc?id=integrate_what_is_api.htm&language=en_US
3. The user must have full access to the Avoka custom objects to deliver submission records into Salesforce. The app package includes permission sets to help manage user access to Avoka Transact objects. Allocate the included Avoka Transact Administrator permission set to this user.
4. The user must also have appropriate permissions to any objects you may wish to update when a submission is delivered as these update functions will be performed as this API user.
5. Ensure the user is licensed to use the Avoka Transact for Salesforce app - How to manage user licensing for the Salesforce App

It is important to understand the impacts of password expiry on integration user accounts. This linked article describes the process of disabling password expiry if you choose to do so:
How to remove password expiry on a Salesforce User Account

Creating the Avoka Transact Connected App in Salesforce

1. Under App Setup, click Create > Apps
2. Under Connected Apps, click New
3. Suggested Connected App attributes:

Attribute	Value
Connected App Name	**Avoka Transact Connector**
API Name	**Avoka_Transact_Connector**
Contact Email	*Suggest using the email address of your Salesforce administrator*

4. Ensure 'Enable OAuth Settings' is checked with the following attributes set:

Attribute	Value
Enable OAuth Settings	**Yes** (selected)
Callback URL	**https://<Transact Domain>/sfdc/secure/account/home.htm** *where <Transact Domain> is the domain of your Avoka Transact environment*
Use Digital Signatures	**No** (unselected)
Selected OAuth Scopes	**Access and manage your data** **Access your basic information** **Provide access to your data via the Web** *It is important to set all and only these 3 scopes*

5. Under 'Web App Settings' enter the following values:

Attribute	Value
Start URL	**https://<Transact Domain>/sfdc/servlet/SmartForm.html?formCode=AT4SF-canvas** *where <Transact Domain> is the domain of your Avoka Transact environment*

Enable SAML	**No** (unselected)

6. Under 'Mobile App Settings' leave blank.
7. Under 'Canvas App Settings' ensure 'Force.com Canvas' is checked and enter the following values:

Attribute	Value
Force.com Canvas	**Yes** (selected)
Canvas App URL	**https://<Transact Domain>/sfdc/servlet/SmartForm.html?formCode=AT4SF-canvas** *where <Transact Domain> is the domain of your Avoka Transact environment*
Access Method	**Signed Request (POST)**
SAML Initiation Method	**None**
Locations (Selected)	**Chatter Tab** **Layouts and Mobile Cards** **Visualforce Page**
Lifecycle Class	*Leave blank*

8. Click save

Configure Permitted Users for the Connected App

The app currently only supports Administrator Approved users.

1. Under 'Administration Setup', click Manage Apps > Connected Apps.
2. Locate the Avoka Transact connected app and click Edit.
3. Under 'OAuth Policies' selected 'Admin approved users are pre-authorized' for Permitted Users.
4. Save.
5. Again under 'Administration Setup', click Manage Apps > Connected Apps locate the Avoka Transact and click the App name link to view the detail page.
6. Under 'Profiles' click Manage Profiles and select the profiles you wish to pre-approve for app usage (typically System Administrator and Standard User).
7. Save.

Creating the Avoka Transact API Named Credential in Salesforce

If your integration scenarios include pushing form task assignments from Salesforce to Avoka in batch or within an Apex trigger (not very common), you will need to setup an API user in Avoka and configure the named credential in Salesforce. The named credential is not required for other scenarios and this step may be skipped.

1. Under Administration Setup, click Security Controls > Named Credentials
2. Under Named Credentials, click New Named Credential
   Enter the following Named Credential attribute values:

Attribute	Value
Label	**Transact API Credential**
Name	**Transact_API_Credential** *must match this exactly*

URL	**https://<Transact Domain>/manager/secure/rest/groovy-service-invoke/v2/at4sf-postrequest/v1/** *where* • *<Transact Domain> is the domain of your Avoka Transact environment* • *the version number at the end (v1) must refer to the version of the AT4SF-PostRequest service in Transact Manager that you wish to call (version 1 at the time of writing).* For older Avoka Transact Manager environment you may need to use the GroovyServiceServlet URL: **https://<Transact Domain>/manager/secure/GroovyServiceServlet?sfmServiceName=AT4SF-PostRequest**
Certificate	*Leave blank unless you have certificate based authentication in place.*
Identity Type	**Named Principal**
Authentication Protocol	**Password Authentication**
Username Password	*The username and password used must correspond to an Avoka Transact user who belongs to the required organization.*

3. Click save

IP White-Listing

Please be aware that if your Transact Manager server has IP white-listing enabled, you will need to add the Salesforce IP ranges to the white-list for the task creation calls to be successful. See:

- https://help.salesforce.com/articleView?id=000003652&type=1
- Post Request: 403 Forbidden - You don't have permission to access /manager/secure/GroovyServiceServlet on this server.

Information required by the Avoka Transact Team

Once the integration User Account and Connected App have been correctly configured in Salesforce, the Avoka Transact administration staff will require the following information:

1. Salesforce Login URL (this will typically be https://login.salesforce.com or https://test.salesforce.com)
2. API User - Username
3. API User - Password
4. API User - Security Token
5. Connected App - Consumer Key
6. Connected App - Consumer Secret

Please review the following articles for instructions on how to retrieve this information from Salesforce:

- How to retrieve the Security Token for a Salesforce User Account
- How to retrieve the Consumer Key and Secret for a Salesforce Connected App

How to enable access to the Avoka Transact app via the App Menu

Unknown macro: 'redirect'

This guide provides instructions on how to give user profiles access to the Avoka Transact App pages via the Salesforce App Menu.

Step-by-step guide

1. Under App Setup, click Create > Apps
2. Under Apps, click 'Edit' next to the Avoka Transact App
3. Under 'Assign to Profiles' on the App Edit Page, select the user profiles that you wish to give access to the App
4. Save

How to manage user licensing for the Salesforce App

Unknown macro: 'redirect'

If your organization has a site license for the Avoka Transact for Salesforce app, all users will automatically be licensed and no license management will be required. This article is relevant where your organization has purchased a limited number of seats.

This guide details how to manage which users are licensed to use the app, where a limited number of user license is available.

Step-by-step guide

1. Under App Setup > Installed Packages, click Manage Licenses next to the Avoka Transact for Salesforce Package
2. Under Licensed Users, add and remove licensed users as required.

How to retrieve the Consumer Key and Secret for a Salesforce Connected App

Unknown macro: 'redirect'

If you're setting up the Salesforce integration using a Connected App, you'll need to get your Consumer Key, Secret and Security Token from Salesforce.

Retrieving the Consumer Key and Secret

1. Under App Setup, click Create > Apps
2. Open the target Connected App and retrieve the Consumer Key and Consumer Secret from the API (oAuth) section
   Note: The Consumer Secret may be protected by a 'Click to reveal' link

How to retrieve the Security Token for a Salesforce User Account

Unknown macro: 'redirect'

Each Salesforce user account has an associated security token that is commonly required when integrating with Salesforce. Your Salesforce security token should have been emailed to you when you set up your Salesforce account *or* the last time you reset your password. If you don't know the security token for a selected user account there is no way to retrieve it through the UI, so you may want to first search for it through old emails to the user's registered email address before resetting the security token.

If the user account is already being used for integration purposes, resetting the security token may break these integrations until they are updated to use the new token.

Resetting a Security Token

Follow this procedure to reset a security token against a user account:

1. Login to Salesforce with the target user account
2. Click on the Username in the top right and select 'Setup' from the menu (for newer environments you may select 'My Settings')
3. Under Personal Setup, click My Personal Information > Reset My Security Token
4. Click the 'Reset Security Token' button
5. Check the user's email inbox for an email from Salesforce advising of the new security token.

Missing 'Reset My Security Token' Option?

See the following troubleshooting article:

- Missing "Reset My Security Token" Option

How to remove password expiry on a Salesforce User Account

Unknown macro: 'redirect'

When setting up an API integration user with Salesforce, a password expiry event will break the integration with little or no warning. This article describes how to disable password expiry for these selected users only and keep your integration channels open.

Creating a 'Password Never Expires' Permission Set

Follow this procedure to create a permission set that will disable password expiry:

1. Under Administration Setup, click Manage Users > Permission Sets
2. Create a new Permission Set with label 'Password Never Expires' and save
3. Edit the System Permissions against this Permission Set, select the 'Password Never Expires' permission and save

Adding the 'Password Never Expires' Permission Set to a User Account

Once you have created the 'Password Never Expires' Permission Set, follow this procedure to add the permission set to a selected user account:

1. Under Administration Setup, click Manage Users > Users
2. Open the target user account and edit Permission Set Assignments
3. Add the 'Password Never Expires' Permission Set and save

To improve security for integration user accounts you could look at one or more of the following strategies:

- Limit permissions to only the objects it needs to access
- Enable IP address restrictions to the IP ranges of the integrating system(s)
- Turn off integration clients (e.g. Outlook)
- Turn off all tabs

How to configure the Transact UI in Salesforce

Unknown macro: 'redirect'

This article describes the process for configuring the Avoka Transact Canvas UI in Salesforce so that users can acces this UI on the object detail pages.

Create the Visualforce Page

A Visualforce page needs to be created for each Salesforce object type you wish to make the Transact UI available for. By way of demonstration, we will outline the process for enabling the UI for the Contact object.

1. Under App Setup, click Develop > Pages
2. Under Visualforce Pages, click New
3. Suggested Page Attributes:

Attribute	Value
Label	Avoka Transact - Contact
Name	Avoka_Transact_Contact
Description	Contains the Avoka Transact view for the Contact object.
	Leave defaults for other options

4. Enter the following Visualforce Markup and click save:

Visualforce Markup

```
<apex:page standardController="Contact">
<!-- Begin Transact App -->
  <div id="transact-container"></div>
  <apex:canvasApp applicationName="Avoka_Transact_Connector"
        containerId="transact-container"
        width="100%"
        border="0"
        scrolling="no"
        height="350px"
        maxHeight="infinite"
        parameters="{
            SObjectId:'{!HTMLENCODE(Contact.Id)}',
            SObjectType:'Contact',
            ContactFirstName:'{!HTMLENCODE(Contact.FirstName)}',
            ContactLastName:'{!HTMLENCODE(Contact.LastName)}',
            ContactEmail:'{!HTMLENCODE(Contact.Email)}',
            EnableUserView:'false',
            EnableObjectView:'true',
            EnableNewForm:'true',
            EnableGlobalSearch:'true'
                }" />
  <!-- End Transact App -->
</apex:page>
```

5. Save.

Visualforce Markup Attributes

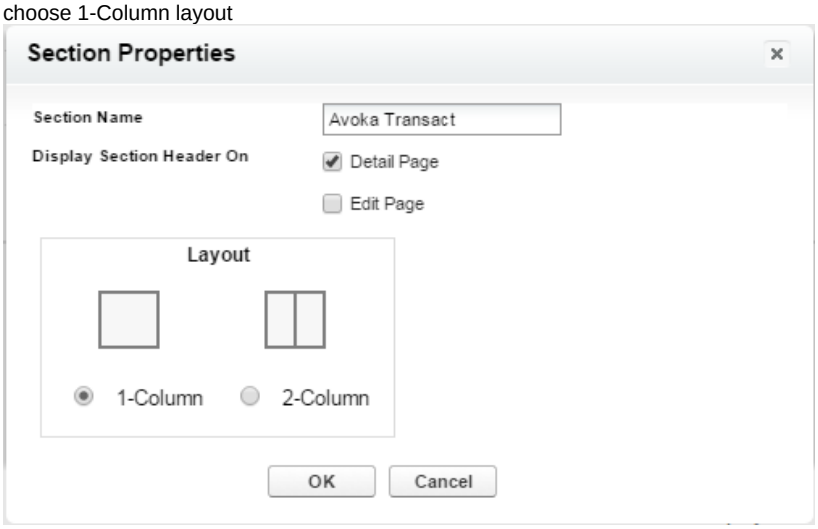
- **standardController="Contact"** - should be set to the name of the object you wish to present the Visualforce page on - in this case 'Contact', if we were creating a Visualforce page for the Account object this value would be 'Account'
- **applicationName="Avoka_Transact_Connector"** - should be set to the name of the Avoka Transct connected app that has been configured - in this case Avoka_Transact_Connector
- **containerId="transact-container"** - set the containerId to the id of the div above the canvas app definition - in this case transact-container
- **width="100%"** - Stretch the canvas app to the full available width
- **border="0"** - Use no border on the frame
- **scrolling="no"** - set scrolling off because we enable the scrollbars in the page layout (see next section)
- **height="350px"** - default height of app
- **maxHeight="infinite"** - app can grow in height unconstrained
- Parameter **SObjectId** - (required) the Id of the Sobject being accessed - in this case just taking the Id of the current Contact record
- Parameter **SObjectType** - (required) a plain english name for the type of the SObject being accessed, may be used in some places throughout the Canvas UI
- Parameter **ContactFirstName** - (optional) The first name of the contact for 'Send to Contact' forms. As we are illustrating with the Contact record we are able to take the Contact's first name, but for other object types we may leave this parameter out

- Parameter **ContactLastName** - (optional) The lastname of the contact for 'Send to Contact' forms. As we are illustrating with the Contact record we are able to take the Contact's last name, but for other object types we may leave this parameter out
- Parameter **ContactEmail** - (optional) The default value to include for the email address of the contact for 'Send to Contact' forms. As we are illustrating with the Contact record we are able to take the Contact's email, but for other object types we may leave this parameter out
- Parameter **EnableUserView** - (optional) A 'true' or 'false' flag to indicate whether to show the User's My Forms page in the Canvas App. Default is false.
- Parameter **EnableObjectView**- (optional) A 'true' or 'false' flag to indicate whether to show the Object Forms page in the Canvas App. Default is false.
- Parameter **EnableNewForm**- (optional) A 'true' or 'false' flag to indicate whether to show the New Form page in the Canvas App. Default is false.
- Parameter **EnableGlobalSearch**- (optional) A 'true' or 'false' flag to indicate whether to show the Search page in the Canvas App. Default is false.
- Parameter **InitialView** - (optional) The name of the page to present when the Canvas App loads. Valid values are 'UserView', 'ObjectView', 'NewForm', 'GlobalSearch'. Default is to initialise on the first included page on the left.
- Parameter **FormCategoryFilter** - (optional) {Version 1.5} A comma separated list of form category names to present in the New Form view. Category names should exactly match the names in Transaction Manager. Without this parameter, all available categories containing forms will be presented.

Note: The HTMLENCODE directive is required for parameter values to avoid unescaped characters.

Add the Visualforce page to the Page Layout

1. Under App Setup, click Customize > Contacts > Page Layouts
2. Under Contact Page Layouts find Contact Layout and click edit
3. In the widget palette at the top, choose Visualforce pages
4. Drag a new Section onto the page in the location you want the Avoka UI to appear, call it Avoka Transact, deselect Edit Page and choose 1-Column layout



5. Drag the Avoka Transact Visualforce page into the section just created and save
6. Edit the Transact Forms properties, set height to 350px and select 'Show Scrollbars'
7. Save.

How to configure Transaction Manager to support the Salesforce App

Unknown macro: 'redirect'

This article describes the process for configuring Avoka Transaction Manager (TM) when integrating with the Salesforce App for the first time.

Prior to commencing this process it is important to confirm that the Salesforce support was included when Transact Manager was installed. The easiest way to check this is to login to TM, navigate to **Forms Form Spaces** and search for the Salesforce space - if the Salesforce space exists then the installation was done correctly.

If the Salesforce support is not installed then this can be easily rectified by submitting a request to the Avoka Cloud Hosting Team (ACS portal) to install the Salesforce support module.

Summary

The process for configuring your TM environment to support the Salesforce App can be summarized as follows:

1. Create the API User Account in TM
2. Import the TM extension packages
3. Configure the imported modules

Creating the API user account in TM

Note this step is only required if you are performing task assignment operations from Salesforce to Avoka Transact. In most cases this will not be the case and so this step can be skipped.

1. Under the 'Security' menu, select 'User Accounts' and click 'New'
2. Under 'User Details' enter the following values:

Attribute	Value
Login Name	**at4sf-client**
User Type	**Local**
Password	*Note the password you use and ensure the 'Change Password After Login' checkbox is unselected*
Given Name(s)	**AT4SF**
Family Name	**Client User**
Email	*Suggest using the email address of your TM or Salesforce administrator*
Portal	**Transaction Manager**

3. Save
4. Under the 'Roles' tab assign the 'Administrator' role to the user and Save
5. Under the 'Organizations' tab assign your organization to the user and Save

Configuring the Salesforce API Client Details

1. Under the 'Services' menu, select 'Service Connections'

2. Locate the Salesforce Service Connection and configure API credentials. Alternatively, for more restricted access you could create a new **Salesforce API Invoke** role with the following minimum permissions:
    a. *Admin Directory*
    b. *Web Service Invoke*
3. Save
4. Update the Security Manager settings as described in 'Configuring the Security Manager' below

Note you will need the connected app and Transact user details from Salesforce to complete this configuration. See How to retrieve the Consumer Key and Secret for a Salesforce Connected App and How to retrieve the Security Token for a Salesforce User Account.

Importing the TM extension packages

See Versions & Installation Assets for access to extension packages.

The following extension packages need to be imported (ideally in this order):

Package	Description
AT4SF-SecurityManager.zip	Contains the OAuth2 Single Sign-On Security Manager Imported under 'Security' > 'Security Managers' > 'Import'
AT4SF-Services.zip	Imported under 'Services' > 'All Services' > 'Import'
AT4SF-Canvas.zip	Contains the Canvas UI that is presented in the Salesforce Desktop environment Imported under 'Forms' > ' Forms' > 'Import Form Version' Select the target organization in the import wizard.
AT4SF-TestForm.zip	Optional test form handy for verifying the connectivity. Imported under 'Forms' > 'Forms' > 'Import Form Version'

Configuring the Salesforce User Space

See the note at the top of this page to ensure the Salesforce support module is installed in the Transact Manager environment.

Now we need to associate the forms with the space

1. In TM navigate to Forms>Organizations
2. Click on your organization
3. On the Spaces tab select the Salesforce space as an 'Assigned Space' and save
4. In TM navigate to Forms>Forms
5. For each of the forms you wish to show under the portal, click on the form name to edit the form
6. On the Spaces tab select the Salesforce space as an 'Assigned Space' and save
7. Don't forget to do this for the Salesforce App form 'AT4SF Canvas'

Configuring the Security Manager

The security manager needs to be configured against the target Salesforce connected app and assigned to the Salesforce User Space as follows:

Note you will need the connected app details from Salesforce to complete this configuration. See How to retrieve the Consumer Key and Secret for a Salesforce Connected App.

1. Under the 'Security' menu in TM, select 'Security Managers'.
2. Locate and open the AT4SF-UserSecurityManager and navigate to the Parameters tab.

3. Edit the 'Client Id' and 'Client Secret' parameters and update to the Consumer Key and Consumer Secret values from the Salesforce connected app (see article linked above).
4. If required, edit the URI parameters to ensure the correct domain name is being used. By default the domain will be the primary Salesforce login domain but for sandbox environments this may need to be changed to test.salesforce.com.
5. If you are using a test Salesforce instance (where the login page is under https://test.salesforce.com/) you will need to update each of the 3 URI parameters accordingly (Auth Uri, Token Uri, User Info Uri).

Parameter	Production Org	Test Org
Auth Uri	https://login.salesforce.com/services/oauth2/authorize	https://test.salesforce.com/services/oauth2/authorize
Token Uri	https://login.salesforce.com/services/oauth2/token	https://test.salesforce.com/services/oauth2/token
User Info Uri	https://login.salesforce.com/services/oauth2/userinfo	https://test.salesforce.com/services/oauth2/userinfo

Ensure the Salesforce User Space is configured to use the correct security manager

1. Under 'Forms' > 'Form Spaces', locate and edit the 'Salesforce' user space.
2. On the 'Space' tab, ensure the 'AT4SF-UserSecurityManager' is selected in the 'Security Manager' field and save.

Configuring the Delivery Service and Delivery Channel

Before delivery into Salesforce can occur the Delivery Service needs to be referenced by an organization Delivery Channel as follows:

1. Under the 'Forms' menu, select 'Organizations'.
2. Locate and open your organization and navigate to the 'Delivery Channels' tab.
3. Click 'New' and enter the following values

Attribute	Value
Name	Salesforce Push Delivery
Delivery Method	Delivery Process
Description	Avoka Transact for Salesforce Generic Push Delivery Channel
Delivery Process	AT4SF-DeliveryPush

4. Save.

Administration Guide - Avoka Transact for Salesforce

The how-to articles in this section detail common administrative tasks for the Salesforce App.

- How to publish forms to Salesforce users
- How to group forms using form categories in TM
- How to configure a form to deliver submissions to Salesforce
- How to configure a Transact Prefill Generator in Salesforce
- How to consume Salesforce prefill data to populate a form
- How to configure a Transact Delivery Processor in Salesforce
- How to restrict access for select forms to certain user groups
- How to enable access for restricted forms to certain user groups
- How to receive administrator alerts when delivery processing fails

How to publish forms to Salesforce users

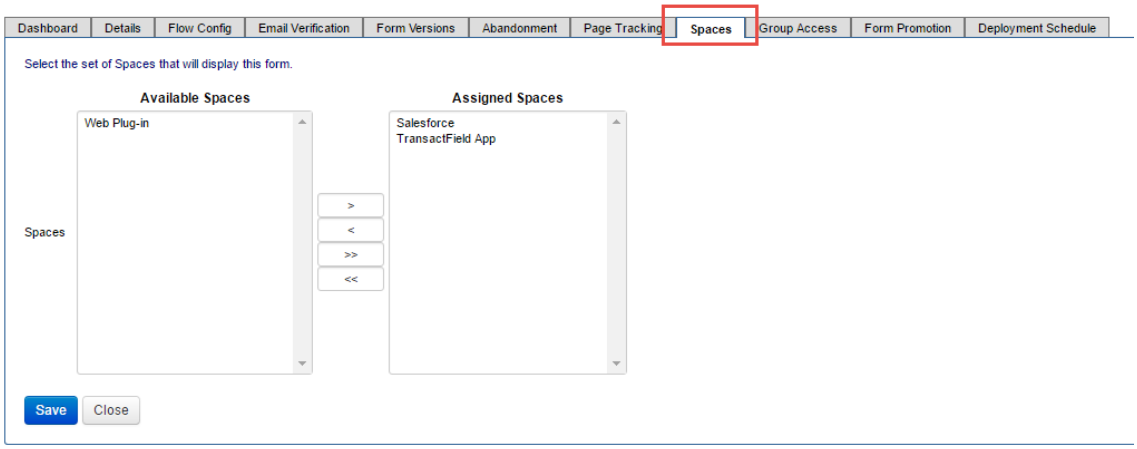
Unknown macro: 'redirect'

To make a form available to Salesforce users it must be assigned into at least one form category and published to the Salesforce User Portal.

Forms that are not published to the Salesforce User Portal or are not assigned into at least one form category will not appear in the Salesforce Desktop UI.

Step-by-step guide

1. Under the 'Forms' menu in TM, select 'Forms'.
2. Find the form you wish to publish to Salesforce users, open it and navigate to the Spaces tab, image below.
3. Assign the form into the Salesforce Space
4. Ensure the form is assigned to at least one category otherwise it will not be accessible from Salesforce. See How to group forms using form categories in TM.



How to group forms using form categories in TM

Unknown macro: 'redirect'

Form categories are used by the Salesforce App to visually group forms together, allowing users to quickly filter down to the forms that are relevant to them.

Step-by-step guide

1. Under the 'Forms' menu in TM select 'Organizations'.
2. Open your organization from the list and navigate to the 'Form Categories' tab.
3. Add the form categories you require by clicking the 'New' button.
4. Under the 'Forms' menu select 'Forms'.
5. For each form that you wish to group, open the active form version and navigate to the 'Form Categories' tab.
6. Assign one or more categories to the form and Save

How to configure a form to deliver submissions to Salesforce

Unknown macro: 'redirect'

Any form that is required to be delivered into Salesforce should be configured with the Salesforce Delivery Channel.

For information on how to configure the Salesforce Delivery Channel, see How to configure Transaction Manager to support the Salesforce App.

Completed Transaction Delivery

1. Under the 'Forms' menu in TM select 'Forms'.
2. Locate and open the form you wish to configure and navigate to the 'Details' tab.
3. Under 'Delivery Channels' select the Salesforce delivery channel (see linked article above) for Production Delivery, Test Delivery or both.
4. Save.

Abandoned Transaction Delivery

1. Under the 'Forms' menu in TM select 'Forms'.
2. Locate and open the form you wish to configure and navigate to the 'Abandonment' tab.
3. Under 'Abandoned Transaction Delivery' select the Salesforce delivery channel (see linked article above) for Abandoned Delivery Channel. For information about the other settings on this page, please consult the Transaction Manager Administration Guide.
4. Save.

Saved Transaction Delivery

1. Under the 'Forms' menu in TM select 'Forms'.
2. Locate and open the form you wish to configure, open the active version record and navigate to the 'Services' tab.
3. Select the AT4SF-SavedProcessor service for Form Saved Processor.
4. Save.

How to configure a Transact Prefill Generator in Salesforce

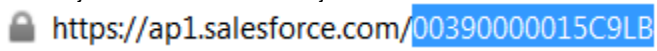
Unknown macro: 'redirect'

This article describes the process for configuring a prefill generator against a particular Transact form in Salesforce. Configuring prefill is a 2 step process:

1. Firstly, identify the data in Salesforce that you wish to use for prefill. This configuration is done in Salesforce by defining a Prefill Generator in the Prefill Generation tab of the Avoka app as described in this article.
2. Secondly, configure where that data will be injected into a form's XML data structure at render time. This configuration is done in the Form Data Config on the Transaction Manager console and is described in this article - How to consume Salesforce prefill data to populate a form.

Create the Transact Prefill Generator Record

1. In the 'App Menu' (top right) select Avoka Transact or simply view All Tabs, then select the Prefill Generation Tab and click new
2. Enter the form code of the Transact form - this value must exactly match the form code in Avoka Transact
3. Retain the default Prefill Generator Class (TransactPrefillGenerator) which allows you to provide JSON configuration to control which fields are included in the prefill generation
4. Specify the Generator Configuration as a JSON string according to the specification defined in this article
5. You may also specify a Sample SObject Id that will allow you to generate a Sample Prefill XML value and test your configuration (Note the SObject Id for a Salesforce object can be obtained from the URL while viewing that object)

6. Save
7. To test the configuration (where you have provided a Sample SObject Id), click the Generate Sample button - the generated prefill XML data will be added to the Sample Prefill XML field if your configuration is correct.
8. You may copy the sample prefill XML and use this in Transaction Manager to define your Input XML prefill mappings for the form.

Managing Form Versions

Where you need to perform different prefill generation between multiple versions of a form you may append an array of versions to the form code attribute to target specific versions. For example, you may wish to have 2 separate prefill generators, one for the first version of the form and another for subsequent versions. This will allow you to configure the prefill generation for new form versions prior to activation of those versions.

To utilize this feature you must specify the specific version numbers as an array directly after the form code. Some samples assuming a form code of 'my-form':

- **my-form**
  Default configuration to be used where no specific configuration is found for the requested version. Shorthand for use of wildcard my-form[*].
- **my-form[1.0]**
  Configuration for version 1.0 of the form. Any versions other than 1.0 will fall back to the default configuration (where no version is specified).
- **my-form[1.0,2.0,3.0]**
  Configuration for versions 1.0, 2.0 and 3.0 of the form.

JSON Configuration Format

This section describes the syntax and supported directives of the JSON configuration consumed by the TransactPrefillGenerator class. This article assumes knowledge of JSON formatting. The JSON configuration should conform to the following structure:

```
{
    "parameters":{
        "<param-name>": "<param-value>"
    },
    "sobjects":{
        "<object-type-name>": {
            "includeFields": ["<field-name>", ... ],
            "includeRelated": ["<relationship-name>", ... ],
            "includeCustom": {"<custom-label>": "<custom-value>", ... }
        },
        ...
    }
}
```

The following example configuration illustrates all supported directives:

```
{
    "parameters":{
        "rootElementName": "SalesforceData"
    },
    "sobjects":{
        "Contact": {
            "includeFields":["Id","FirstName","LastName","Email","Phone","Title"],
            "includeRelated":["Account"],
            "includeCustom":{"FullName": "{!FirstName} {!LastName}"}
        },
        "Account": {
            "includeFields":["Name","Type","Industry"],
            "includeRelated":["Owner","Opportunities"]
        },
        "Opportunity": {
            "includeFields":["Name","Amount"]
        },
        "User": {
            "includeFields":["Name","Email","Username","Manager.Name"]
        }
    }
}
```

Supported Parameters

Parameter Name	Description
rootElementName	(Optional) The name to be given to the root element in the generated XML prefill data. Defaults to **SFDCPrefillData**.

Object Type Definition

A single JSON configuration can support any number of Salesforce object types, thereby allowing the form to be prefilled with many different object types. For example, the same form may be prefilled from a contact record or a lead record.

The configuration for an object type must be contained in an attribute labeled with the name of that object type. Note the configuration below for the Account object:

```
"Account": {
    "includeFields":["Name","Type","Industry"]
}
```

Dealing with Complex Types

If you wish to prefill data from Salesforce that is contained in complex types (e.g. Addresses), you should configure the individual data sub-attributes in your prefill configuration. For example if you want to prefill the Shipping Address from the Account object, your configuration should be similar to this:

```
{
    "sobjects":{
        "Account": {
            "includeFields":["Name","ShippingStreet","ShippingCity","ShippingState","ShippingPostalCode","ShippingCountry"]
        }
    }
}
```

The includeFields Directive (Required)

The **includeFields** directive is required for all object prefill definitions and allows you to explicitly specify the fields to be included in the prefill data with the following considerations:

- The field list should be provided as a comma separated list of string litterals where each item specifies the field name, not the label of the field.
- Parent and referential relationships can be leveraged by using the 'dot' notation to traverse the relationships by the name of the relationship (E.g. "includeFields": "FirstName,LastName,Owner.Name"). Note: if a child relationship is specified it will be ignored, but child relationships are supported in the includeRelated directive.
- Include the Id field if you want to support form user edits to the object information. If the Id value is present when the competed submission is processed on its return to Salesforce, the delivery processor may use this Id to update the object according to the updates made in the f

The includeRelated Directive (Optional)

The **includeRelated** directive is optional and allows you to specify related objects to include in the prefill data with the following considerations:

- Specifying a related entity requires that the object type definition exists in the JSON configuration for that related entity.
- The object type definition must specify the name of the object type, not the name of the relationship. As an example, consider the directive "includeRelated":"Owner". The relationship name is 'Owner' but the related entity is a 'User' so the JSON configuration must contain an object definition named 'User'.
- Child relationships are supported, provided that the object is not itself a child of the primary prefill entity.

The includeCustom Directive (Optional)

The **includeCustom** directive is optional an allows you to specify custom fields to include in the prefill data with the following considerations:

- Hard coded values may be used. E.g. "includeCustom":{"CampaignCode":"FF102"}
- Field references to the object context may be used but the referenced fields must be specified in the includeFields directive. E.g. "includeCustom":{"FullName": "{!FirstName} {!LastName}"}

XML Prefill Data Generation

The TransactPrefillGenerator interprets the JSON configuration and generates the XML data used to prefill the form according to the following rules:

- XML root element is always named 'SFDCPrefillData' unless an alternative root element name is specified.
- Each sub-node within the root element represents a single Salesforce object or a list of objects
- All fields specified in the **includeFields** or includeCustom directives will be included in the object node
- Where a related field is specified in the **includeFields** directive (E.g. Manager.Name), that field will generated the parent object node in a containing element named according to the relationship
- Where the **includeRelated** directive includes a parent or referencial relationship (E.g. "includeRelated":"Owner"), the object node for the related object will be added to the XML root node and named according to the relationship, e.g.

```
<Owner>
    <Name>Jeff Johnson</Name>
    <Manager>
        <Name>Jenny Quack</Name>
    </Manager>
</Owner>
```

- Where the **includeRelated** directive includes a child relationship, a list container node will be added to the XML root node and named according to the relationship, each object node within the container will be named with the object type name. E.g.

```
<Opportunities>
    <Opportunity>
        <Name>Chicago City Store Displays</Name>
        <Amount>$50,000.00</Amount>
    </Opportunity>
    <Opportunity>
        <Name>San Francisco Mobile Signage</Name>
        <Amount>$28,000.00</Amount>
    </Opportunity>
</Opportunities>
```

The following block illustrates an example of the XML prefill data produced by the generation engine:

```
<?xml version="1.0" encoding="UTF-8"?>
<SFDCPrefillData>
    <Contact>
        <Id>00390000015C9LBAA0</Id>
        <FirstName>Kristen</FirstName>
        <LastName>Akin</LastName>
        <Title>Director, Warehouse Mgmt</Title>
        <Phone>(434) 555-3100</Phone>
        <Email>kakin@sony.com</Email>
        <FullName>Kristen Akin</FullName>
    </Contact>
    <Account>
        <Name>Volley Music</Name>
        <Type>Customer - Direct</Type>
        <Industry>Retail</Industry>
    </Account>
    <Owner>
        <Name>Jeff Johnson</Name>
        <Email>jj@domain.com</Email>
        <Username>jj@domain.com</Username>
        <Manager>
            <Name>Jenny Quack</Name>
        </Manager>
    </Owner>
    <Opportunities>
        <Opportunity>
            <Name>Chicago City Store Displays</Name>
            <Amount>$50,000.00</Amount>
        </Opportunity>
        <Opportunity>
            <Name>San Francisco Mobile Signage</Name>
            <Amount>$28,000.00</Amount>
        </Opportunity>
    </Opportunities>
    <User>
        <Name>Johny Lighning</Name>
        <Email>jl@domain.com</Email>
        <Username>jl@domain.com</Username>
        <Manager>
            <Name>Jack Sparrow</Name>
        </Manager>
    </User>
</SFDCPrefillData>
```

Custom Prefill Generators

Where the default prefill generator does not provide the level of control required, you may define your own prefill generator by developing an APEX class that implements the **ITransactPrefillGenerator** interface. To utilise your custom prefill generator, simply specify the class name in the relevant Transact Prefill Generator record and any configuration required by the class.

ITransactPrefillGenerator

```
global interface ITransactPrefillGenerator {

    /**
     * This function generates the prefill XML data based on the config provided.
     *
     * @param prefillObjectId The Id of the object to use for prefill generation
     * @param formPrefillConfig The string containing the generation config
     * @return The generated prefill XML data
     */
    String generatePrefillXml(Id entityId, String config);
}
```

How to consume Salesforce prefill data to populate a form

Unknown macro: 'redirect'

Configuration of form pre-population from Salesforce generated prefill data is managed by the standard Input XML Prefill Mappings in the form data config.

For instructions on how to configure prefill data generation in Salesforce, see How to configure a Transact Prefill Generator in Salesforce. This article also describes how to generate sample prefill data for use in the mapping configuration.

Step-by-step guide

1. Under the 'Forms' menu in TM, select 'Forms'.
2. Locate and open the form you wish to pre-populate.
3. From the Dashboard tab open the 'Data Config' page of the active form version and navigate to the 'Input XML Prefill Mapping' tab.
4. Click 'New Input XML' and past the Sample Prefill XML from the Salesforce Prefill Generator (see article linked above), enter a version name and save.
5. Click 'Edit XML Mapping' and map each input data element with the target element in the form XML.
6. Save and close.

Note that Input XML Prefill Mappings do not accommodate mapping for repeating elements, so if you are generating prefill data that has multiple records of information in it (e.g. all cases related to a contact) there is currently no simple process available out-of-the-box to map this into the form and a more customized solution will need to be assessed.

How to configure a Transact Delivery Processor in Salesforce

Unknown macro: 'redirect'

This article describes the process for configuring a delivery processing function against a particular Transact form in Salesforce, so that you can update your standard and custom Salesforce objects with data from the submitted form. The consideration as to which object to create/update in Salesforce is typically unique to the use case being targeted but often this involves the Contact or Lead objects.

For information about the Salesforce standard objects you can view their own documentation. E.g. Salesforce Sales Objects

Create the Transact Delivery Processor Record

1. In the 'App Menu' (top right) select Avoka Transact or simply view All Tabs, then select the Delivery Processing Tab and click new
2. Enter the form code of the Transact form - this value must exactly match the form code in Avoka Transact
3. Retain the default Delivery Processor Class (TransactDeliveryProcessor) which allows you to provide JSON configuration to control what objects are affected by the submission and how
4. Specify the Processing Configuration as a JSON string according to the specification defined in this article
5. Save

Delivery Trigger Types

Delivery records are received from Avoka Transact according to the triggers specified in Transaction Manager. Transactions are typically delivered when completed, but they can also be delivered when the transaction has been saved or deemed to be abandoned, to facilitate lead generation and follow up.

Managing Form Versions

Where you need to perform different delivery processing functions between multiple versions of a form you may append an array of versions to the form code attribute to target specific versions. For example, you may wish to have 2 separate delivery processing functions, one for the first version of the form and another for subsequent versions. This will allow you to continue to accept deliveries for in-flight transactions on old form versions after a new version is activated.

To utilize this feature you must specify the specific version numbers as an array directly after the form code. Some samples assuming a form code of 'my-form':

- **my-form**
  Default configuration to be used where no specific configuration is found for the requested version. Shorthand for use of wildcard my-form[*].
- **my-form[1.0]**
  Configuration for version 1.0 of the form. Any versions other than 1.0 will fall back to the default configuration (where no version is specified).
- **my-form[1.0,2.0,3.0]**
  Configuration for versions 1.0, 2.0 and 3.0 of the form.

JSON Configuration Format

This section describes the syntax and supported directives of the JSON configuration consumed by the TransactPrefillGenerator class. This article assumes knowledge of JSON formatting. The JSON configuration should conform to the following structure:

```
{
    "parameters":{
        "<param-name>": "<param-value>"
    },
    "sobjects":{
        "<object-label>": {
            "type": "<object-api-name>",
            "matchFields": ["<field-name>", ... ],
            "insertFields": ["<field-name>", ... ],
            "updateFields": ["<field-name>", ... ],
            "sourceFields": {
                "<field-name>": "<field-mapping>",
                ...
            },
            "relations": {
                "<relationship-name>": "<object-label>",
                ...
            }
        },
        ...
    }
}
```

The following example configuration illustrates all supported directives:

```
{
    "parameters": {
        "formAssociationTarget":"new-case",
        "attachmentTarget":"new-case",
        "receiptTarget":"new-case",
        "caseNumber":"new-case.CaseNumber"
    },
    "sobjects": {
        "parent-account": {
            "type": "Account",
            "insertFields":[],
            "updateFields":[],
            "sourceFields" : {
                "Id":"[!XML://Account/AccountId]"
            }
        },
        "my-contact": {
            "type": "Contact",
            "matchFields":["LastName","Email"],
            "insertFields":["*"],
            "updateFields":["Email","Phone","MobilePhone","Title","Department"],
            "sourceFields" : {
                "Id":"[!XML://Contact/ContactId]",
                "Salutation":"[!XML://Contact/Title]",
                "FirstName":"[!XML://Contact/FirstName]",
                "LastName":"[!XML://Contact/LastName]",
                "Phone":"[!XML://Contact/Phone]",
                "MobilePhone":"[!XML://Contact/Mobile]",
                "Email":"[!XML://Contact/Email]",
                "Title":"[!XML://Contact/Position]",
                "Department":"[!XML://Contact/Department]"
            },
            "relations": {
                "Account": "parent-account"
            }
        },
        "new-case": {
            "type": "Case",
            "insertFields":["*"],
            "updateFields":["*"],
            "sourceFields" : {
                "Type":"[!XML://IssueDetails/IssueType]",
                "Reason":"[!XML://IssueDetails/Reason]",
                "Subject":"[!XML://IssueDetails/Summary]",
                "Description":"[!XML://IssueDetails/Description]",
                "Origin":"Web"
            },
            "relations": {
                "Account": "parent-account",
                "Contact": "my-contact"
            }
        }
    }
}
```

Supported Parameters

Parameter Name	Description
formAssociationTarget	(Optional) The object (specified by referencing the relevant <object-label>) to associate the Transact form with so that it appears against that object in the Canvas App.

attachmentTarget	(Optional) The object (specified by referencing the relevant <object-label>) to add the submitted attachments to.
receiptTarget	(Optional) The object (specified by referencing the relevant <object-label>) to add the generated PDF receipt to.
caseNumber	(Optional) The field (specified by referencing the <object-label>.<field-name>) from which to retrieve the Salesforce case number. This case number is returned to Avoka Transact and stored against the submission.

Object Type Definition

A single JSON configuration can define impacts to any number of Salesforce objects.

Each object configuration must be given a label that is unique in the configuration. Note the configuration below for a Contact object:

```
"my-contact": {
    "type": "Contact",
    "matchFields":["LastName","Email"],
    "insertFields":["*"],
    "updateFields":["Email","Phone","MobilePhone","Title","Department"],
    "sourceFields" : {
        "Id":"[!XML://Contact/ContactId]",
        "Salutation":"[!XML://Contact/Title]",
        "FirstName":"[!XML://Contact/FirstName]",
        "LastName":"[!XML://Contact/LastName]",
        "Phone":"[!XML://Contact/Phone]",
        "MobilePhone":"[!XML://Contact/Mobile]",
        "Email":"[!XML://Contact/Email]",
        "Title":"[!XML://Contact/Position]",
        "Department":"[!XML://Contact/Department]"
    },
    "relations": {
        "Account": "parent-account"
    }
}
```

The matchFields Directive (Optional)

{Version 1.5} The matchFields directive provides a basic de-duplication function by first searching for a matching record before determining whether to perform an insert or update. To utilize this optional function, provide an array of Salesforce field names that should be used to identify an existing record. If a record is found in Salesforce where ALL the listed fields are an exact match for the information provided in the form, this record will be updated.

- The field list should be provided as a comma separated list with no spaces or new lines where each item specifies the field name, not the label of the field.

For example, the following directive will cause the delivery processor to search for a Contact object with matching LastName and Email:

```
"my-contact": {
    "type": "Contact",
    "matchFields":["LastName","Email"],
    ...
```

The insertFields Directive (Required)

The **insertFields** directive is required for all object definitions and allows you to explicitly define which fields will be used in an insert scenario (when an Id value is not available).

- The field list should be provided as a comma separated list with no spaces or new lines where each item specifies the field name, not the label of the field.
- The field list should not include the "Id" field.
- A wild card value of "*" may be used to denote that all sourced fields should be affected in the insert.

The updateFields Directive (Required)

The **updateFields** directive is required for all object definitions and allows you to explicitly define which fields will be used in an update scenario (when an Id value is available).

- The field list should be provided as a comma separated list with no spaces or new lines where each item specifies the field name, not the label of the field.

- There is no need to include the "Id" field - it will be added automatically if it is available.
- A wild card value of "*" may be used to denote that all sourced fields should be affected in the update.

The sourceFields Directive (Required)

The **sourcefields** directive is required for all object definitions and allows you to define the mappings from the submitted XML data into the object fields.

```
"sourceFields" : {
    "Id":"[!XML://Contact/ContactId]",
    "Salutation":"[!XML://Contact/Title]",
    "FirstName":"[!XML://Contact/FirstName]",
    "LastName":"[!XML://Contact/LastName]",
    ....
}
```

- Hard-coded values may be used instead of XML mappings where appropriate. E.g.:

  ```
  "Origin":"Web"
  ```

- XML mappings can be specified such that values from the XML submission data are injected into the fields at processing time. XML mappings must be structured as follows:

  ```
  [!XML://<xml location>]
  ```

  where <xml location> is the URI to the XML element containing the required value. Note the root XML node is not specified in the location.
  If your data contains repeating nodes than you may target a specific node index to pull data from a specific node by specifying the node index (starting at zero) in parentheses. For example the following mapping will pull the Email data element from the 2nd Applicant node in the XML:

  ```
  [!XML://Applicants/Applicant(1)/Email]
  ```

- {Version 1.8} Lookup fields can be populated with record Ids of related objects by specifying a LOOKUP function that must be structured as follows:

  ```
  [!LOOKUP://<object-name>?<field-name>=<field-value>[&<field-name>=<field-value>]
  ```

  For example, the following lookup function will search for a Contact record with matching first name and last name and insert the resulting record Id into the Contact__c lookup field:

  ```
  "Contact__c":"[!LOOKUP://Contact?FirstName=[!XML://Contact/FirstName]&LastName=[!XML://Contact/LastName]]"
  ```

  The value will be left empty if no matching record is found. If multiple records are found, an exception will result.

The relations Directive (Optional)

The **relations** directive is optional and allows you to specify object relationships to other objects defined in the configuration.

```
"relations": {
    "Account": "parent-account",
    "Contact": "my-contact"
}
```

- For each relationship specified, the key value should match the name of a relationship against the object and the reference value should match an object label of the correct type specified in the configuration.

XML Consumption

The example configuration above will consume the following XML submission data structure to create the new Case object and make updates to the contact information for the Contact:

```
<?xml version="1.0" encoding="UTF-8"?>
<AvokaSmartForm>
    <Account>
        <AccountId>00128000002gH6DAAU</AccountId>
        <AccountName>Big Top Enterprises</AccountName>
        <Industry>Entertainment</Industry>
        <Website>www.bigtop.com</Website>
        <AccountRep>Ben Warner</AccountRep>
    </Account>
    <Contact>
        <ContactId>00328000001tCzaAAE</ContactId>
        <Title>Mr.</Title>
        <FirstName>Jolly</FirstName>
        <LastName>Roger</LastName>
        <Position>CEO</Position>
        <Department>Management</Department>
        <Phone>+61 2 9955 6600</Phone>
        <Mobile>0411 803 240</Mobile>
        <Email>jolly@bigtop.com</Email>
    </Contact>
    <IssueDetails>
        <IssueType>Problem</IssueType>
        <Reason>Instructions not clear</Reason>
        <Summary>Can't find where screw C3 goes</Summary>
        <Description>It is holding together but for how long?</Description>
    </IssueDetails>
</AvokaSmartForm>
```

Custom Delivery Processors

Where the default delivery processor does not provide the level of control required, you may define your own delivery processor by developing an APEX class that implements the ITransactDeliveryProcessor interface. To utilise your custom delivery processor, simply specify the class name in the relevant Transact Delivery Processor record and any configuration required by the class.

ITransactDeliveryProcessor

```
global interface ITransactDeliveryProcessor {

    /**
     * This function processes the delivery record based on the config provided.
     *
     * @param delivery The Transact Delivery object to be processed
     * @param processorConfig The string containing the processing config
     */
    void processDelivery(Transact_Delivery__c delivery, String config);
}
```

By way of example, the following custom delivery processor class is configured to send an email to an inbox when the submission is delivered to Salesforce.

Custom Delivery Processor Example

```
global class CustomDeliveryProcessor implements avoka.ITransactDeliveryProcessor {

    public void processDelivery(avoka__Transact_Delivery__c delivery, String processorConfig){
        System.debug('Custom Delivery Processor: ' + delivery.Name);

        Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
        String[] toAddresses = new String[] {'salesforce@avoka.com'};
        mail.setToAddresses(toAddresses);
        mail.setSubject('Custom Delivery Processor: ' + delivery.Name);
        mail.setPlainTextBody('Processing complete: ' + delivery.Name);
        Messaging.sendEmail(new Messaging.SingleEmailMessage[] { mail });
    }
}
```

To configure your custom delivery processor to be used you must enter the class name in the Delivery Processor object with the appropriate form code. The default value is 'TransactDeliveryProcessor' - you can modify this to be your own class name and it will be used instead of the default. You can also specify your own configuration data if required - your processor class will receive this configuration in the 'config' string parameter in the processDelivery() method signature.

If you have any trouble getting your custom delivery processor class to be used, please use the 'global' keyword in the class declaration to ensure it is visible to the Avoka managed package.

How to restrict access for select forms to certain user groups

Unknown macro: 'redirect'

This article describes the process for configuring access restrictions for selected forms to specific user groups.

By default, access to forms is unrestricted within the security context of the form. To restrict access for select form to certain user groups we need to first create a form group and then allocate users and forms into that group.

Create the Form Group

1. In Avoka Transaction Manager, select the 'Security' menu, then select 'Groups'.
2. Under 'Groups' click the 'New' button.
3. Enter the name that you'd like to call the group, relevant to the type of users who will be members of the group.
4. For 'Type' select 'Form'.
5. Optionally enter a description and select the required group user access control settings. Its recommended to enable all settings unless there are specific reasons not to.
6. Save.

Add Users to the Form Group

Users will not have access to forms in form group until they are added to the form group.

1. In Avoka Transaction Manager, select the 'Security' menu, then select 'Groups'.
2. Under 'Groups', select the group you wish to manage.
3. Select the 'Members' tab and add users to the group as required.
4. Save.

Assign Forms to the Form Group

1. In Avoka Transaction Manager, select the 'Forms' menu, then select 'Forms'.
2. Under 'Forms', select the form you wish to manage.
3. Select the 'Group Access' tab and assign the form into the available groups as required.
4. Save.

How to enable access for restricted forms to certain user groups

Unknown macro: 'redirect'

This article describes how Salesforce Administrators enable access for restricted forms to certain user groups within Salesforce.

This article assumes that the restricted form groups have already been established in Avoka Transaction Manager.

See: How to restrict access for select forms to certain user groups

Create the Public Group

In order to enable access to restricted form groups, a Salesforce Public Group must be created with the identical name to the Form Group in Avoka Transaction Manager.

1. Under 'Administration Setup', select 'Manage Users', then 'Public Groups'.
2. Under 'Public Groups', click 'New'.
3. Enter the label and name of the group, ensuring that the Label is identical to associated Form Group name in Avoka Transaction Manager.
4. Configure the membership to the group.
5. Save.

Once created, users who are members of the public group will automatically gain access to the restricted forms the next time they load the Canvas app.

How to receive administrator alerts when delivery processing fails

Unknown macro: 'redirect'

This article describes how to activate administrator alerts to one or more users, when a delivery processing failure occurs.

Step-by-step guide

When delivery processing for a particular transaction fails, it will be identified in the Delivery Log with a status of 'Failed' and failure reason will be available in the Processing Message field.

Delivery processing failure alerts are sent out to anyone who has the **Avoka Transact Administrator** permission set on their account. To add this permission set to a user account:

1. Under Administration Setup > Manage Users > Users, locate and open the user account.
2. Under Permission Set Assignments click 'Edit Assignments'
3. Move the 'Avoka Transact Administrator' permission set from the 'Available' to the 'Enabled' column.
4. Save.

Troubleshooting

Review the Error Log

If something is not working in the integration between Avoka and Salesforce, it pays to start by reviewing the **Error Log** in transaction manager (found under the **System** menu). This may quickly reveal the cause of the issue.

- Post Request: 403 Forbidden - You don't have permission to access /manager/secure/GroovyServiceServlet on this server.
- Content cannot be displayed: You do not have sufficient privileges to access the page: /apex/Avoka_Transact_Lead
- Missing "Reset My Security Token" Option
- Insufficient Privileges: You do not have the level of access necessary to perform the operation you requested...
- Delivery Error: No such column 'avoka__Delivery_Trigger_Type__c' on sobject of type avoka__Transact_Delivery__c
- Delivery Error: The requested resource does not exist
- Delivery Processing Error: Field <X> on <Y> is not CREATEABLE by <Z>
- Could not identify SObject type <X>
- Sorry, you are not licensed to use the Avoka Transact solution.
- Salesforce API Authentication Failure (Invalid Grant)

Content cannot be displayed: You do not have sufficient privileges to access the page: /apex/Avoka_Transact_Lead

Unknown macro: 'redirect'

Problem

Some of your users are getting a message displayed when they try to access the Avoka visualforce page in a page layout:

- *Content cannot be displayed: You do not have sufficient privileges to access the page: /apex/Avoka_ Transact_Lead*

This issue may occur if you have enabled security around the visualforce page that does not include the profile of the user trying to access the page.

Solution

To resolve this issue you must enable page access to the users profile:

1. Under Setup > Develop, select Visualforce Pages
2. Under Visualforce Pages click the 'Security' link next to the target page
3. Add the profiles that require access to the page and save

Reference

- https://help.salesforce.com/apex/HTViewHelpDoc?id=pages_security_page_def.htm&language=en_US

Could not identify SObject type <X>

Unknown macro: 'redirect'

Problem

When processing a delivery an exception is thrown with error message:

> *Could not identify SObject type <X>*

where <X> is an object type specified in your delivery processor configuration.

Solution

This issue indicates that there is a problem with the delivery processor configuration. Please check that you have specified all the SObject types correctly in your configuration.

Delivery Error: No such column 'avoka__Delivery_Trigger_Type__c' on sobject of type avoka__Transact_Delivery__c

Unknown macro: 'redirect'

Problem

The Delivery Process in Transaction Manager fails with error message:

> ***SalesForceException[response=[{ "message" : "No such column 'avoka__Delivery_Trigger_Type__c' on sobject of type avoka__Transact_Delivery__c", "errorCode" : "INVALID_FIELD" }], operation=perform***

This issue relates to insufficient permissions on the Salesforce side and is caused by missing permissions in the *Avoka Transact Administrator* per mission set prior to version 1.10 of the in the Avoka Transact for Salesforce app.

The *Avoka Transact Administrator* permission set should have full permissions on the 4 Avoka objects (Delivery Log, Delivery Processing, Post Log, Prefill Generation). In app versions prior to 1.10 you will notice a number of missing permissions when you view the object settings of this packaged permission set. The screenshot below shows that some field permissions are not activated:

Field Permissions

Field Name	Read	Edit
Affected Sobject Id	☑	☑
Attachment Count	☑	☐
Attachment Metadata	☐	☐
Case Number	☑	☑
Created By	☑	☐
Data Extracts	☐	☐
Delivery Trigger Type	☐	☐
Form Code	☑	☑
Form Name	☐	☐
Form Version	☐	☐
Job Reference Code	☐	☐
Last Modified By	☑	☐
Owner	☑	☑
Payment Amount	☑	☐
Payment Gateway Receipt Number	☑	☐
Payment Gateway Timestamp	☑	☐
Payment Gateway Transaction Number	☑	☐
Portal Name	☐	☐
Processed Timestamp	☑	☐
Processing Message	☑	☑
Processing Status	☑	☐
Submission Data	☑	☑
Submission Key	☑	☑
Submission Timestamp	☑	☐
Sync Status	☐	☐
Tracking Code	☑	☑
Transaction Score	☑	☑
Transaction Status	☐	☐
User Name	☑	☑

Solution

This issue is only applicable to versions of the Avoka Transact for Salesforce App prior to 1.10. The permission set in version 1.10 has the missing permissions added.

As the Avoka Transact Administrator permission set is part of a managed package, it cannot be modified in your org. To resolve this issue, you will need to create a new permission set that has all permissions enabled for the 4 Avoka objects.

1. Under Administer, click Manage Users > Permission Sets
2. Under Permission Sets, click New and enter the label of the new permission set

Attribute	Value
Label	Avoka Transact Administrator 1
Description	Used to activate missing permissions in the Avoka Transact Administrator permission set
User License	<blank>

3. Click Save
4. Edit the Object Settings of the permission set to activate all object and field permissions for the 4 Avoka objects:
    a. Delivery Log
    b. Delivery Processing
    c. Prefill Generation
    d. Post Log
5. Ensure all read and edit permissions are granted to these objects.
6. Save.
7. Lastly, allocate this new permission set to the Avoka Transact user.

Delivery Error: The requested resource does not exist

Unknown macro: 'redirect'

Problem

The Delivery Process in Transaction Manager fails with error message:

> ***SalesForceException[response=[{ "errorCode" : "NOT_FOUND", "message" : "The requested resource does not exist" }]***

Solution

This error indicates that the delivery process cannot find the Delivery target objects on the Salesforce side. To resolve this issue:

1. Ensure that the app is installed in Salesforce as per the instructions - see How to configure Salesforce for integration with Avoka Transact
2. If the license is limited to a number of seats then the Avoka Transact API user must be licensed for the app in order to access the Avoka delivery objects - see How to manage user licensing for the Salesforce App.

Delivery Processing Error: Field <X> on <Y> is not CREATEABLE by <Z>

Unknown macro: 'redirect'

Problem

When processing a delivery an error is reported with message:

> ***Delivery Processing Error: Field <X> on <Y> is not CREATEABLE by <Z>***

where X is the target object name, Y is the field being updated and Z is the user name.

The same error may also appear with the key word **UPDATEABLE**.

Solution

This is a permission related problem and relates to the security around field updates on the object. Ensure the active user specified in the error message has the appropriate permissions to the target field.

Insufficient Privileges: You do not have the level of access necessary to perform the operation you requested...

Unknown macro: 'redirect'

Problem

While trying to access the Avoka Transact Visualforce page component some users are denied access with the following message:

> ***Insufficient Privileges***
>
> *You do not have the level of access necessary to perform the operation you requested. Please contact the owner of the record or your administrator if access is necessary. For more information, see Insufficient Privileges Errors.*

This problem may be encountered if the package was installed for access by Admins Only or Selected Profiles instead of for All Users.

Solution

To resolve this issue please add the **Avoka Transact User** permission set to all required user accounts:

1. In the Setup menu search for Permission Sets and select the menu item to open the Permission Sets page.
2. Find the **Avoka Transact User** permission set and open it, then select the Manage Assignments button.
3. Select Add Assignments and assign all required users to the permission set.

Missing "Reset My Security Token" Option

Unknown macro: 'redirect'

Problem

After creating the Avoka Transact user in Salesforce, no Security Token was emailed out and and when logged in as that user the option to 'Reset My Security Token is not available as described in this article:

- How to retrieve the Security Token for a Salesforce User Account

Solution

If you do not see the option to reset your token, you may be using IP Login Restrictions in Salesforce. This Salesforce article describes why this menu option may not be available to your user:

- https://help.salesforce.com/HTViewSolution?id=000182450

If you have further questions about IP Login Restrictions, please contact Salesforce support for further assistance.

Post Request: 403 Forbidden - You don't have permission to access /manager/secure/GroovyServiceServlet on this server.

Unknown macro: 'redirect'

Problem

When attempting to push a task from Salesforce to Avoka Transact Manager using the Post Request object in Salesforce you may receive an error indicating that the HTTP request failed, e.g.

> 403 Forbidden - You don't have permission to access /manager/secure/GroovyServiceServlet on this server.

This may be due to IP white-listing on the Avoka server rejecting access from the Salesforce server.

Solution

Add the Salesforce IP ranges to the Avoka Transact Manager IP while-list:

- https://help.salesforce.com/articleView?id=000003652&type=1

Reference

Salesforce API Authentication Failure (Invalid Grant)

Unknown macro: 'redirect'

Problem

Calls to Salesforce from Transaction Manager are failing with an authentication error:

```
SalesForceException[response={"error_description":"authentication failure","error":"invalid_grant"}
```

If you are receiving Delivery Error when trying to deliver to Salesforce, check the error log for the error message above.

Solution

This issue can be caused due to one of the following reasons:

- Invalid or incorrect credentials being used in the Salesforce Connection parameters
- Insufficient privileges assigned to the API user in Salesforce

To resolve this issue, try the following steps:

1. Re-enter the username, password and security token for the Salesforce Service Connection. Note if you do not have a record of the security token, please see this article on how to regenerate it - How to retrieve the Security Token for a Salesforce User Account.
2. Ensure that the Salesforce API user has the required permissions. This article describes how to configure the Avoka Transact user - How to configure Salesforce for integration with Avoka Transact.
    a. The user must have API access.
    b. The user must have permission to write to the Avoka custom objects.
    c. Ensure the user is associated with a profile that is Admin approved for use of the Avoka Transact connected app.
3. If you have IP Restrictions in place in your Salesforce environment, double check that the IP ranges cover the Transact Manager environment you are receiving the error on. You can check with the Avoka cloud hosting team to verify the IP ranges for your environments.

Sorry, you are not licensed to use the Avoka Transact solution.

Unknown macro: 'redirect'

Problem

Trying to access the Avoka Transact Visualforce page component and getting a message saying:

> ***Sorry, you are not licensed to use the Avoka Transact solution. Please consult your Salesforce Administrator.***

Solution

Your user is not licensed for use of the package. A Salesforce administrator needs to add you as a licensed user as described in the article below:

- How to manage user licensing for the Salesforce App

Versions & Installation Assets

Unknown macro: 'redirect'

Any new deployment should be using the latest available version compatible with your Avoka Transact instance. Please review the Minimum Manager Version requirement against your installed version of Transaction Manager.

Title	Version	Release Date	Description	Manager Version
Version 1.11	1.11	August 24, 2017	Resolved issue with encoding of Post Request data	4.3
Version 1.10	1.10	March 7th, 2016	Point release to accommodate customer driven features and bug fixes	4.3
Version 1.8	1.08	December 16th, 2015	Point release to accommodate customer driven features and bug fixes	4.3
Version 1.6	1.06	December 2nd, 2015	Minor patch release to improve permissions around Apex class access. No functional changes are included.	4.3
Version 1.5	1.05	November 28th, 2015	Point release to provide client requested features and bug fixes.	4.3
Version 1.4	1.04	September 23rd, 2015	This is the original AppExchange version containing baseline features.	4.2

Salesforce Portal War

Manager Version	Portal WAR File
4.3.4+	From 4.3.4 the Salesforce portal war file is built into the Manager installer.
4.3.3	avoka-sf-portal-salesforce433.war
4.3.2	avoka-sf-portal-salesforce432.war

Version 1.4

Unknown macro: 'redirect'

Description	This is the original AppExchange version containing baseline features.
Version	1.04
Release Date	September 23rd, 2015
Manager Version	4.2

This release version has been superseded. Please install the latest version of the App.

Installation Assets

The Salesforce App can be installed into the Salesforce Organization by clicking this link and following the prompts:

- https://login.salesforce.com/packaging/installPackage.apexp?p0=04t900000002ZZ8

Version 1.5

Unknown macro: 'redirect'

Description	Point release to provide client requested features and bug fixes.
Version	1.05
Release Date	November 28th, 2015
Manager Version	4.3

New Features

- SAE-085 Match Fields - You can now optionally specify the set of unique fields for an object type (e.g. LastName and Email for a Lead) and the delivery processor will find the existing record for update rather than insert. This feature was added to assist in the management of duplicate records.
- SAE-151 Administrator Alerts - Anyone in with the ‘Avoka Transact Administrator’ permission set in Salesforce will receive email notifications when delivery processing fails.
- SAE-055 Form Category Filter - In the visual force page to include the Canvas UI you can now optionally specify which categories you want to appear on the ‘New Form’ page. This can be different for each instance of the Canvas. If not specified, all categories will be presented.
- SAE-146 Abandoned Forms – Abandoned forms will now appear in the search results (requires Transaction Manager 4.3). When a form becomes abandoned, customers cannot access it anymore. We've added a Reactivate link that is presented for abandoned forms where the data has not yet been purged. The reactivate link will re-open the form for access.
- SAE-147 Enhanced Customer Link - Provided the ability to copy a portal specific URL for an active form by selecting the portal from a drop down. This is to cater for the situation where customers interact with a different portal to staff.
- SAE-152 Reload page button - On all UI pages other than Search, the reload button will refresh the content of the page.

Issues Resolved

- SAE-135 Authentication Provider Issues - Resolved issues where existing users were not getting added to the current organization and portal when they re-authenticate under certain conditions.
- SAE-148 Incorrect Created Time - Resolved issue where the created time of a form was showing incorrect values under certain conditions.

Installation Assets

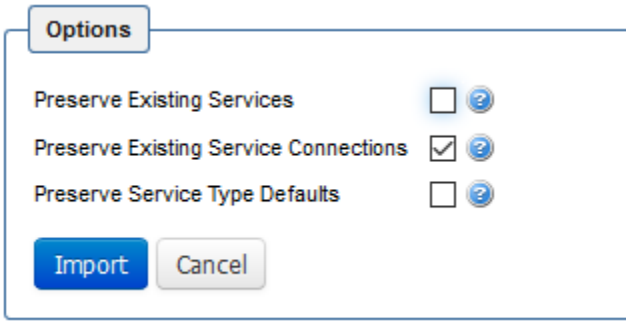
The Salesforce App can be installed into the Salesforce Organization by clicking this link and following the prompts:

- https://login.salesforce.com/packaging/installPackage.apexp?p0=04t900000002ZjX

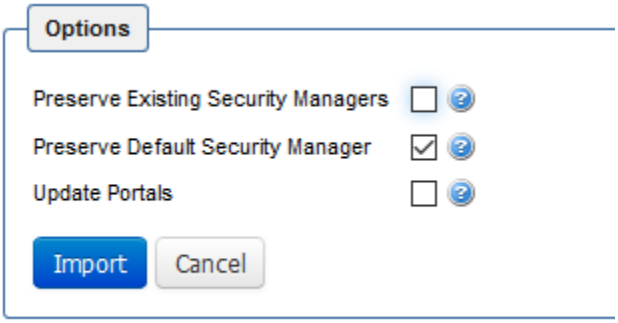
The following installation archives should be used for importing assets Transaction Manager:

- Service Definitions: AT4SF-Services-1.5.zip
- Security Manager: AT4SF-SecurityManager-1.5.zip
- Canvas UI Form Version: AT4SF-Canvas-1.5.zip
- Test Form Version: AT4SF-TestForm-1.0.zip

When importing service definitions deselect the 'Preserve Existing Services' option in the import wizard while leaving the 'Preserve Existing Service Connections' option selected:



Similarly, when importing the security manager deselect the 'Preserve Existing Security Managers' option:



Note that importing the security manager in this way will overwrite the parameters you have specified against this security manager so please make a backup copy of them first. In particular, the Client Id and Client Secret properties should be backed up and re-set after import.

Version 1.6

Unknown macro: 'redirect'

Description	Minor patch release to improve permissions around Apex class access. No functional changes are included.
Version	1.06
Release Date	December 2nd, 2015
Manager Version	4.3

Installation Assets

The Salesforce App can be installed into the Salesforce Organization by clicking this link and following the prompts:

- https://login.salesforce.com/packaging/installPackage.apexp?p0=04t90000000Y1N0

No change to the Transaction Manager installation assets since Version 1.5.

Version 1.8

Unknown macro: 'redirect'

Description	Point release to accommodate customer driven features and bug fixes
Version	1.08
Release Date	December 16th, 2015
Manager Version	4.3

New Features

- SAE-166 Lookup Fields - Add support for the mapping of form data to lookup field references

Issues Resolved

- SAE-161 Relationship between the transaction and the current object not created where no prefill service is specified for the form
- SAE-163 Exception thrown if Delivery Processor not configured for specified form code
- SAE-164 Null Pointer if SObjectType not found
- SAE-167 Exception in delivery mapping to Boolean, Decimal or Date fields

Installation Assets

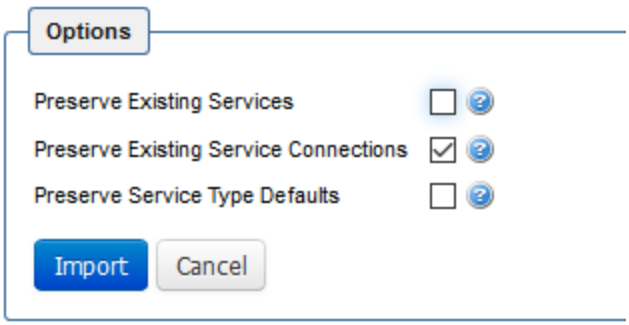
The Salesforce App can be installed into the Salesforce Organization by clicking this link and following the prompts:

- https://login.salesforce.com/packaging/installPackage.apexp?p0=04t90000000Y1SP

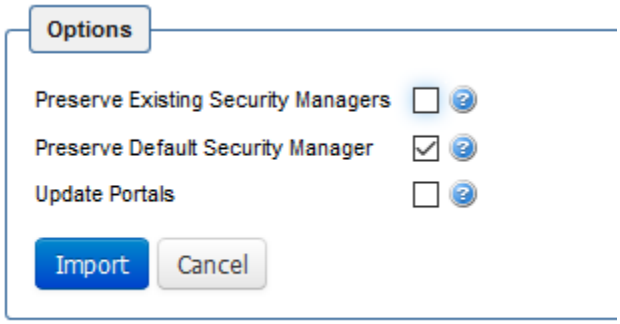
The following installation archives should be used for importing assets Transaction Manager:

- Service Definitions: AT4SF-Services-1.8.zip
- Security Manager: AT4SF-SecurityManager-1.5.zip
- Canvas UI Form Version: AT4SF-Canvas-1.5.zip
- Test Form Version: AT4SF-TestForm-1.0.zip

When importing service definitions deselect the 'Preserve Existing Services' option in the import wizard while leaving the 'Preserve Existing Service Connections' option selected:



Similarly, when importing the security manager deselect the 'Preserve Existing Security Managers' option:



Note that importing the security manager in this way will overwrite the parameters you have specified against this security manager so please make a backup copy of them first. In particular, the Client Id and Client Secret properties should be backed up and re-set after import.

Version 1.10

Unknown macro: 'redirect'

Description	Point release to accommodate customer driven features and bug fixes
Version	1.10
Release Date	March 7th, 2016
Manager Version	4.3

New Features

- SAE-173 - Support for date time fields in delivery mapping
- SAE-175 - New Preprocessor service option
- SAE-176 - New Completed Processor service option

Issues Resolved

- SAE-168 - Exception if relationship parent is not found in delivery processing
- SAE-169 - Auto role assignment not working in Auth Provider due to error code
- SAE-174 - Manual Delivery Processing ignores trigger type (always executes Completed Transaction processing)

Installation Assets

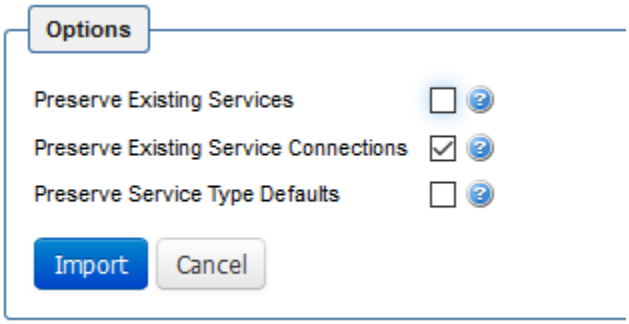
The Salesforce App can be installed into the Salesforce Organization by clicking this link and following the prompts:

- Production: https://login.salesforce.com/packaging/installPackage.apexp?p0=04t90000000Y1cA
- Sandbox: https://test.salesforce.com/packaging/installPackage.apexp?p0=04t90000000Y1cA

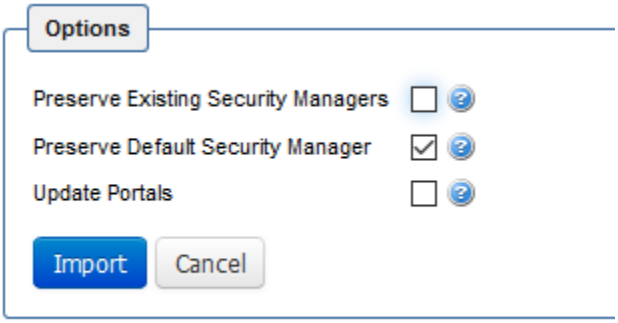
The following installation archives should be used to setup Transaction Manager:

- Service Definitions: AT4SF-Services-1.10.zip
  - New service AT4SF-PushToSalesforce - performs submission delivery to Salesforce and is used as a sub process by other services
  - New service AT4SF-CompletedProcessor - provides a service option for delivery to Salesforce on the Submission Completed event
  - New service AT4SF-Preprocessor - provides a service option for delivery to Salesforce on the Submission Preprocessor event
  - Updated service AT4SF-DeliveryPush (v3) - now offloads delivery process to AT4SF-PushToSalesforce
  - Updated service AT4SF-SaveProcessor (v2) - now offloads delivery process to AT4SF-PushToSalesforce
- Security Manager: AT4SF-SecurityManager-1.10.zip
- Canvas UI Form Version: AT4SF-Canvas-1.5.zip
- Test Form Version: AT4SF-TestForm-1.0.zip

When importing service definitions deselect the 'Preserve Existing Services' option in the import wizard while leaving the 'Preserve Existing Service Connections' option selected:



Similarly, when importing the security manager deselect the 'Preserve Existing Security Managers' option:



Note that importing the security manager in this way will overwrite the parameters you have specified against this security manager so please make a backup copy of them first. In particular, the Client Id and Client Secret properties should be backed up and re-set after import.

Note: once you have imported new versions of the services you are required to manually activate these service versions and where appropriate, update the configurations that refer to the old version of the service to point to the new version.

Version 1.11

Unknown macro: 'redirect'

Description	Resolved issue with encoding of Post Request data
Version	1.11
Release Date	August 24, 2017
Manager Version	4.3

New Features

None.

Issues Resolved

- SAE-179 Post log fails if data contains '&' symbol - When using the Post Log with prefill data from Salesforce objects that contains special symbols like '&' (e.g. "Walker & Sons") the post request content is corrupted.

Installation Assets

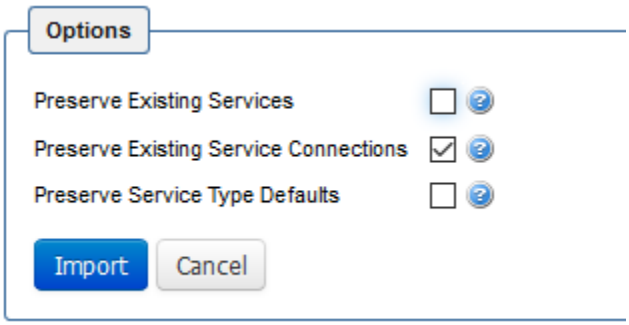
The Salesforce App can be installed into the Salesforce Organization by clicking this link and following the prompts:

- Production: https://login.salesforce.com/packaging/installPackage.apexp?p0=04t90000000JDR9
- Sandbox: https://test.salesforce.com/packaging/installPackage.apexp?p0=04t90000000JDR9

The following installation archives should be used to setup Transaction Manager:

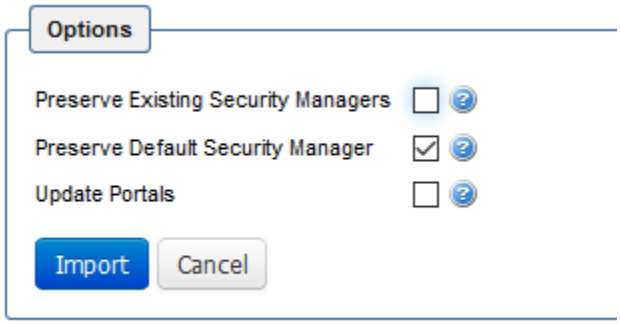
- Service Definitions: AT4SF-Services-1.10.zip
- Security Manager: AT4SF-SecurityManager-1.11.zip
- Canvas UI Form Version: AT4SF-Canvas-1.5.zip
- Test Form Version: AT4SF-TestForm-1.0.zip

When importing service definitions deselect the 'Preserve Existing Services' option in the import wizard while leaving the 'Preserve Existing Service Connections' option selected:



Note: once you have imported new versions of the services you are required to manually activate these service versions and where appropriate, update the configurations that refer to the old version of the service to point to the new version.

Similarly, when importing the security manager deselect the 'Preserve Existing Security Managers' option:



Note that importing the security manager in this way will overwrite the parameters you have specified against this security manager so please make a backup copy of them first. In particular, the Client Id and Client Secret properties should be backed up and re-set after import.

Note: once you have imported new versions of the services you are required to manually activate these service versions and where appropriate, update the configurations that refer to the old version of the service to point to the new version.

How-to articles

Add how-to article

Title	Creator	Modified
How to manage user licensing for the Salesforce App	Unknown User (bwarner)	May 15, 2019
How to configure Transaction Manager to support the Salesforce App	Unknown User (bwarner)	May 15, 2019
How to receive administrator alerts when delivery processing fails	Unknown User (bwarner)	May 15, 2019
How to enable access for restricted forms to certain user groups	Unknown User (bwarner)	May 15, 2019
How to restrict access for select forms to certain user groups	Unknown User (bwarner)	May 15, 2019
How to configure a Transact Delivery Processor in Salesforce	Unknown User (bwarner)	May 15, 2019
How to consume Salesforce prefill data to populate a form	Unknown User (bwarner)	May 15, 2019
How to configure a Transact Prefill Generator in Salesforce	Unknown User (bwarner)	May 15, 2019
How to configure a form to deliver submissions to Salesforce	Unknown User (bwarner)	May 15, 2019
How to group forms using form categories in TM	Unknown User (bwarner)	May 15, 2019
How to configure the Transact UI in Salesforce	Unknown User (bwarner)	May 15, 2019
How to enable access to the Avoka Transact app via the App Menu	Unknown User (bwarner)	May 15, 2019
How to publish forms to Salesforce users	Unknown User (bwarner)	May 14, 2019

Troubleshooting articles

Add troubleshooting article

Title	Creator	Modified
Sorry, you are not licensed to use the Avoka Transact solution.	Unknown User (bwarner)	May 15, 2019
Salesforce API Authentication Failure (Invalid Grant)	Unknown User (bwarner)	May 15, 2019
Missing "Reset My Security Token" Option	Unknown User (bwarner)	May 15, 2019
Insufficient Privileges: You do not have the level of access necessary to perform the operation you requested...	Unknown User (bwarner)	May 15, 2019
Delivery Processing Error: Field <X> on <Y> is not CREATEABLE by <Z>	Unknown User (bwarner)	May 15, 2019
Delivery Error: No such column 'avoka__Delivery_Trigger_Type__c' on sobject of type avoka__Transact_Delivery__c	Unknown User (bwarner)	May 15, 2019
Could not identify SObject type <X>	Unknown User (bwarner)	May 15, 2019
Content cannot be displayed: You do not have sufficient privileges to access the page: /apex /Avoka_Transact_Lead	Unknown User (bwarner)	May 15, 2019
Delivery Error: The requested resource does not exist	Unknown User (bwarner)	May 15, 2019