Informatica® Cloud Data Integration

REST V2 Connector

Informatica Cloud Data Integration REST V2 Connector
April 2020

Publication Date: 2020-05-17

Table of Contents

Preface

Use *REST V2 Connector* to learn how to read from or write to a web service that supports REST API by using Cloud Data Integration. Learn to create a REST V2 connection, develop and run mappings and mapping tasks in Cloud Data Integration.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Intelligent Cloud Services web site

You can access the Informatica Intelligent Cloud Services web site at http://www.informatica.com/cloud. This site contains information about Informatica Cloud integration services.

Informatica Intelligent Cloud Services Communities

Use the Informatica Intelligent Cloud Services Community to discuss and resolve technical issues. You can also find technical tips, documentation updates, and answers to frequently asked questions.

Access the Informatica Intelligent Cloud Services Community at:

https://network.informatica.com/community/informatica-network/products/cloud-integration

Developers can learn more and share tips at the Cloud Developer community:

https://network.informatica.com/community/informatica-network/products/cloud-integration/cloud-developers

Informatica Intelligent Cloud Services Marketplace

Visit the Informatica Marketplace to try and buy Data Integration Connectors, templates, and mapplets:

https://marketplace.informatica.com/

Data Integration connector documentation

You can access documentation for Data Integration Connectors at the Documentation Portal. To explore the Documentation Portal, visit https://docs.informatica.com.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Intelligent Cloud Services Trust Center

The Informatica Intelligent Cloud Services Trust Center provides information about Informatica security policies and real-time system availability.

You can access the trust center at https://www.informatica.com/trust-center.html.

Subscribe to the Informatica Intelligent Cloud Services Trust Center to receive upgrade, maintenance, and incident notifications. The Informatica Intelligent Cloud Services Status page displays the production status of all the Informatica cloud products. All maintenance updates are posted to this page, and during an outage, it will have the most current information. To ensure you are notified of updates and outages, you can subscribe to receive updates for a single component or all Informatica Intelligent Cloud Services components. Subscribing to all components is the best way to be certain you never miss an update.

To subscribe, go to https://status.informatica.com/ and click **SUBSCRIBE TO UPDATES**. You can then choose to receive notifications sent as emails, SMS text messages, webhooks, RSS feeds, or any combination of the four.

Informatica Global Customer Support

You can contact a Customer Support Center by telephone or online.

For online support, click **Submit Support Request** in Informatica Intelligent Cloud Services. You can also use Online Support to log a case. Online Support requires a login. You can request a login at https://network.informatica.com/welcome.

The telephone numbers for Informatica Global Customer Support are available from the Informatica web site at https://www.informatica.com/services-and-training/support-services/contact-us.html.

CHAPTER 1

Introduction to REST V2 Connector

This chapter includes the following topics:

REST V2 Connector overview

Use REST V2 Connector to interact with web service applications that support REST API. You can use REST V2 Connector in a Source transformation, Target transformation, or midstream in a Web Services transformation.

You can use REST V2 Connector midstream in a mapping to pass a single or multiple requests to a web service application and process the response data. You can also pass data obtained from multiple transformations in the mapping pipeline and process the data.

When you use REST V2 Connector midstream in a mapping, you first create a business service for the operation that you want to perform in the web service application. You then associate the business service in a Web Services transformation midstream in a mapping to read from or write data to the web service application. For example, you can use REST V2 Connector as a midstream transformation to perform PUT operation on a web service application.

You can use special characters in field names for target and midstream transformations.

You can use the following REST methods in source, target, and midstream transformation:

- GET
- PUT
- POST
- DELETE
- OPTIONS
- HEAD

When you can create a REST V2 connection, you can configure one of the following REST authentication types:

- Standard
  - BASIC
  - DIGEST
  - OAuth 1.0
- OAuth 2.0 client credentials
- OAuth 2.0 authorization code

You can use the following media types to process data:

- application/xml
- application/json
- application/x-www-form-urlencoded
- JSON subtype. For example: `application/**hal+**json`
- JSON custom type. For example: `application/**vnd.ds.abc.v1+**json`
- Extended JSON mime type. For example: `application/vnd.ds.abc.v1+json**;version=q1**`
- text/xml

REST V2 Connector task and object types

The following table lists the REST V2 transformation types that you can include in Data Integration tasks:

Task Type	Transformation Type
Mapping	Source, Target, and Midstream

When you run a mapping or mapping task, the session log is saved to the following directory: `<Secure Agent installation directory>/apps/Data_Integration_Server/logs`.

XML objects

REST V2 Connector supports XML objects such as, XML attributes, the wrapper object, and the namespace object.

You can define attributes only for XML objects. You cannot define attributes for XML elements in the swagger specification file. Wrapper objects are supported for a simple type array element only when the array element has inline definitions. REST V2 Connector supports qualified and unqualified namespace objects in Response for source, midstream, and target transformations. Whereas, REQUEST with qualified and unqualified namespace objects is supported for midstream and target transformations.

Swagger definition for an XML attribute:

```
"XMLAttrArray_Request##body##Employee##dep##Mangr" : {
```

```
"properties" : {
"desg" : {
"type" : "string",
"xml": { "attribute": true }
}, "mid" : { "type" : "string" },
```

The XML attribute sample:

```
<dep>
<Mangr desg="Director">
<mid>em09</mid>
<mname>mg</mname>
</Mangr>
<id>did</id>
<name>dname</name>
</dep>
```

An XML attribute is always of type string. Even if the swagger definition has attribute with type as number, the REST V2 connector always treats the attribute as string. The XSD generated has the data type as string for the XML attribute.

Swagger definition for a Wrapper object:

```
"books" : {
"type" : "array",
"items" : { "type" : "string" },
"xml": {
"wrapped" : "true", "name": "books-array"
}
}
```

The Wrapper object sample:

```
<books-array>
<books>1</books>
<books>2</books>
</books-array>
```

Swagger definition for a Namespace object:

```
root" : {
"properties" : {
"table" : {
"$ref" : "#/definitions/NSResReq_Request##body##root##table"
}
},
```

```
"xml":{
"prefix": "h",
"namespace": "http://www.w3.org/TR/html4/"
}
}
```

The namespace object sample:

```
<h:root xmlns:h="http://www.w3.org/TR/html4/">
<table>
<tr>
<td>Apples</td>
<td>Bananas</td>
</tr>
</table>
</h:root>
```

Administration of REST V2 Connector

As a user, you can use REST V2 Connector after the organization administrator performs the following tasks:

- Installs the Secure Agent on a 64-bit machine.
- Ensures that the machine hosting the Secure Agent has a minimum memory size of 2048 MB.
- Deploys the latest version of the XToolkit API to the Secure Agent directory.
- Installs the Data Transformation package on the Secure Agent machine.
- Installs the UDT Notation Generator package on the Secure Agent machine.
- Provides the license to the Data Transformation package, UDT Notation Generator package, and the saas-xmetadataread package on the Secure Agent machine.

Secure communication

You can configure TLS authentication to establish one-way or two-way secure communication with the REST API.

The Secure Agent establishes a secure connection with the REST API over TLS. You can use one-way SSL or two-way SSL.

Use One-Way SSL

To use one-way SSL, perform the following steps:

- Import the server certificate to the `<Secure Agent installation directory>\jre\lib\security\cacerts` file. Use the following command:
  `keytool -importkeystore -srckeystore <PathtoCert>\clientSSL.p12 -srcstoretype <type of certificate pkcs12> -destkeystore <Informatica agent Installation location\jdk\jre\lib\security\cacerts -deststoretype JKS`

- Add JVM options for truststore file name and truststore password:
  - Click Administrator > Runtime Environments and select an agent.
  - Select Type as DTM under **System Configuration Details**.
  - Add the following JVM options:
    - `JVMOption1=-Djavax.net.ssl.trustStore=<absolute path of the .jks truststore file>`
    - `JVMOption2=-Djavax.net.ssl.trustStorePassword=<truststore password>`

  You can also specify the name of the keystore file and keystore password in the **KeyStore File Name** and **KeyStore Password** connection properties.

Use Two-Way SSL

To use two-way SSL, you must first perform the steps for one-way SSL, and then perform the following steps:

- Add JVM options for keystore file and keystore password:
  - Click Administrator > Runtime Environments and select an agent.
  - Select Type as DTM under **System Configuration Details**.
  - Set the following JVM options:
    - `JVMOption3=-Djavax.net.ssl.keyStore=<absolute path of the .jks keystore file>`
    - `JVMOption4=-Djavax.net.ssl.keyStorePassword=<keystore password>`

  You can also specify the name of the keystore file and keystore password in the **KeyStore File Name** and **KeyStore Password** connection properties.

The Secure Agent processes the certificate in the following order:

Keystore

1. Connection attributes
2. JVM property

Truststore

1. Connection attributes
2. JVM property
3. Certificate imported at `<Secure Agent installation directory>\jdk\jre\lib\security\cacerts`

Limitations on Hosted Agent

Consider the following limitations when you run mappings on the Hosted Agent:

- The swagger specification file URL must be a public URL and must return the content of the file without prompting for further authentication and redirection.
- The REST APIs that require custom server certificate signed by CA, which are not a part of Informatica cacerts truststore, are not supported on the Hosted Agent.
- JWT bearer token authentication is not supported on the Hosted Agent.

CHAPTER 2

REST V2 connections

This chapter includes the following topics:

REST V2 connection overview

Create a REST V2 connection to make calls to a web service application.

When you create a connection, specify the swagger specification file and the authentication method if required. You can specify the following authentication methods:

- Standard
- OAuth 2.0 client credentials
- OAuth 2.0 authorization code
- JWT bearer token

REST V2 Connector supports swagger specification version 2.0. The swagger specification file contains operation ID, path parameters, query parameters, header fields, request details, and response details.

You create a REST V2 connection on the **Connections** page.

If your REST endpoint does not have a swagger specification, you can generate the swagger specification file from Administrator. Click Administrator > Swagger Files to generate a swagger specification file.

For information about parameters to create a swagger file, see *Data Integration Connections*.

REST V2 connection properties

When you set up a REST V2 connection, you must configure the connection properties.

The following table describes the REST V2 connection properties for a standard authentication type connection:

Connection property	Description
Authentication Type	If required, select the authentication method that the connector must use to login to the web service application. Default is none.
Auth User ID	The user name to login to the web service application. Required only for Basic and Digest authentication types.
Auth Password	The password associated with the user name. Required only for Basic and Digest authentication types.
OAuth Consumer Key	The client key associated with the web service application. Required only for OAuth authentication type.
OAuth Consumer Secret	The client password to connect to the web service application. Required only for OAuth authentication type.
OAuth Token	The access token to connect to the web service application. Required only for OAuth authentication type.
OAuth Token Secret	The password associated with the OAuth token. Required only for OAuth authentication type.
Swagger File Path	The absolute path along with the file name or the hosted URL of the swagger specification file. If you are providing the absolute path of the swagger specification file, the swagger specification file must be located on the machine that hosts the Secure Agent. **Note:** You can generate the swagger specification file from Administrator. Click Administrator > Swagger Files to generate a swagger specification file.
TrustStore File Path	The absolute path of the truststore file that contains the TLS certificate to establish a one-way or two-way secure connection with the REST API. Specify a directory path that is available on each Secure Agent machine in the runtime environment. You can also configure the truststore file name and password as a JVM option or import the certificate to the following directory: `<Secure Agent installation directory\jre\lib\security\cacerts.`
TrustStore Password	The password for the truststore file that contains the SSL certificate. You can also configure the truststore password as a JVM option.
KeyStore File Name	The absolute path of the keystore file that contains the keys and certificates required to establish a two-way secure communication with the REST API. Specify a directory path that is available on each Secure Agent machine in the runtime environment. You can also configure the keystore file name and location as a JVM option or import the certificate to any directory.
KeyStore Password	The password for the keystore file required for secure communication. You can also configure the keystore password as a JVM option.

Connection property	Description
Proxy Type	Type of proxy. You can select one of the following options: - No Proxy: Bypasses the proxy server configured at the agent or the connection level. - Platform Proxy: Proxy configured at the agent level is considered. - Custom Proxy: Proxy configured at the connection level is considered.
Proxy Configuration	The proxy configuration format: `<host>:<port>` **Note:** Proxy server with authentication is not supported.
Advanced Fields	Enter the arguments that the Secure Agent uses when connecting to a REST endpoint. You can specify the following arguments, each separated by a semicolon (;): `ConnectionTimeout`: the wait time in milliseconds to get a response from a REST endpoint. The connection ends after the connection timeout is over. Default is the timeout defined in the endpoint API. **Note:** If you define both the REST V2 connection timeout and the endpoint API timeout, the connection ends at the shortest defined timeout. `connectiondelaytime`: the delay time in milliseconds to send a request to a REST endpoint. Default is 10000. `retryattempts`: number of times the connection is attempted. Default is 3. `qualifiedSchema`: specifies if the schema selected is qualified or unqualified. Default is false. Example: `connectiondelaytime:10000;retryattempts:5`

OAuth 2.0 client credentials authentication

The following table describes the REST V2 connection properties for an OAuth 2.0 - Client Credentials authentication type connection:

Connection property	Description
Access Token URL	Access token URL configured in your application.
Client ID	Client ID of your application.
Client Secret	Client secret of your application.
Scope	Specifies access control if the API endpoint has defined custom scopes. Enter space separated scope attributes. For example: `root_readonly root_readwrite manage_app_users`
Access Token Parameters	Additional parameters to use with the access token URL. Parameters must be defined in the JSON format. For example: `[{"Name":"resource","Value":"https://<serverName>"}]`
Generate Access Token	Generates access token based on the information provided in the above fields.
Access Token	Enter the access token value or click **Generate Access Token** to populate the access token value.

Connection property	Description
Swagger File Path	The absolute path along with the file name or the hosted URL of the swagger specification file. If you are providing the absolute path of the swagger specification file, the swagger specification file must be located on the machine that hosts the Secure Agent. **Note:** You can generate the swagger specification file from Administrator. Click Administrator > Swagger Files to generate a swagger specification file.
TrustStore File Path	The absolute path of the truststore file that contains the TLS certificate to establish a one-way or two-way secure connection with the REST API. Specify a directory path that is available on each Secure Agent machine in the runtime environment. You can also configure the truststore file name and password as a JVM option or import the certificate to the following directory: `<Secure Agent installation directory\jre\lib\security\cacerts.`
TrustStore Password	The password for the truststore file that contains the SSL certificate. You can also configure the truststore password as a JVM option.
KeyStore File Name	The absolute path of the keystore file that contains the keys and certificates required to establish a two-way secure communication with the REST API. Specify a directory path that is available on each Secure Agent machine in the runtime environment. You can also configure the keystore file name and location as a JVM option or import the certificate to any directory.
KeyStore Password	The password for the keystore file required for secure communication. You can also configure the keystore password as a JVM option.
Proxy Type	Type of proxy. You can select one of the following options: - No Proxy: Bypasses the proxy server configured at the agent or the connection level. - Platform Proxy: Proxy configured at the agent level is considered. - Custom Proxy: Proxy configured at the connection level is considered.
Proxy Configuration	The proxy configuration format: `<host>:<port>` **Note:** Proxy server with authentication is not supported.
Advanced Fields	Enter the arguments that the Secure Agent uses when connecting to a REST endpoint. You can specify the following arguments, each separated by a semicolon (;): `ConnectionTimeout`: the wait time in milliseconds to get a response from a REST endpoint. The connection ends after the connection timeout is over. Default is the timeout defined in the endpoint API. **Note:** If you define both the REST V2 connection timeout and the endpoint API timeout, the connection ends at the shortest defined timeout. `connectiondelaytime`: the delay time in milliseconds to send a request to a REST endpoint. Default is 10000. `retryattempts`: number of times the connection is attempted. Default is 3. `qualifiedSchema`: specifies if the schema selected is qualified or unqualified. Default is false. Example: `connectiondelaytime:10000;retryattempts:5`

OAuth 2.0 authorization code authentication

To use authorization code authentication, you must first register the following Informatica redirect URL in your application:

`https://<Informatica cloud hosting facility for your organization>/ma/proxy/oauthcallback`

If the access token expires, Informatica redirect URL, which is outside the customer firewall, tries to connect to the endpoint and retrieve a new access token.

The following table describes the REST V2 connection properties for an OAuth 2.0 - Authorization Code authentication type connection:

Connection property	Description
Authorization Token URL	Authorization server URL configured in your application.
Access Token URL	Access token URL configured in your application.
Client ID	Client ID of your application.
Client Secret	Client secret of your application.
Scope	Specifies access control if the API endpoint has defined custom scopes. Enter space separated scope attributes. For example: `root_readonly root_readwrite manage_app_users`
Access Token Parameters	Additional parameters to use with the access token URL. Parameters must be defined in the JSON format. For example: `[{"Name":"resource","Value":"https://<serverName>"}]`
Authorization Code Parameters	Additional parameters to use with the authorization token URL. Parameters must be defined in the JSON format. For example: `[{"Name":"max_age","Value":60},{"Name":"state","Value":"test"}]`
Generate Access Token	Generates access token and refresh token based on the information provided in the above fields.
Access Token	Enter the access token value or click **Generate Access Token** to populate the access token value.
Refresh Token	Enter the refresh token value or click **Generate Access Token** to populate the refresh token value. If the access token is not valid or expires, the Secure Agent fetches a new access token with the help of refresh token. **Note:** If the refresh token expires, you must either provide a valid refresh token or regenerate a new refresh token by clicking **Generate Access Token**.
Swagger File Path	The absolute path along with the file name or the hosted URL of the swagger specification file. If you are providing the absolute path of the swagger specification file, the swagger specification file must be located on the machine that hosts the Secure Agent. **Note:** You can generate the swagger specification file from Administrator. Click Administrator > Swagger Files to generate a swagger specification file.
TrustStore File Path	The absolute path of the truststore file that contains the TLS certificate to establish a one-way or two-way secure connection with the REST API. Specify a directory path that is available on each Secure Agent machine in the runtime environment. You can also configure the truststore file name and password as a JVM option or import the certificate to the following directory: `<Secure Agent installation directory\jre\lib\security\cacerts.`

Connection property	Description
TrustStore Password	The password for the truststore file that contains the SSL certificate. You can also configure the truststore password as a JVM option.
KeyStore File Name	The absolute path of the keystore file that contains the keys and certificates required to establish a two-way secure communication with the REST API. Specify a directory path that is available on each Secure Agent machine in the runtime environment. You can also configure the keystore file name and location as a JVM option or import the certificate to any directory.
KeyStore Password	The password for the keystore file required for secure communication. You can also configure the keystore password as a JVM option.
Proxy Type	Type of proxy. You can select one of the following options: - No Proxy: Bypasses the proxy server configured at the agent or the connection level. - Platform Proxy: Proxy configured at the agent level is considered. - Custom Proxy: Proxy configured at the connection level is considered.
Proxy Configuration	The proxy configuration format: `<host>:<port>` **Note:** Proxy server with authentication is not supported.
Advanced Fields	Enter the arguments that the Secure Agent uses when connecting to a REST endpoint. You can specify the following arguments, each separated by a semicolon (`;`): `ConnectionTimeout`: the wait time in milliseconds to get a response from a REST endpoint. The connection ends after the connection timeout is over. Default is the timeout defined in the endpoint API. **Note:** If you define both the REST V2 connection timeout and the endpoint API timeout, the connection ends at the shortest defined timeout. `connectiondelaytime`: the delay time in milliseconds to send a request to a REST endpoint. Default is 10000. `retryattempts`: number of times the connection is attempted. Default is 3. `qualifiedSchema`: specifies if the schema selected is qualified or unqualified. Default is false. Example: `connectiondelaytime:10000;retryattempts:5`

JWT bearer token authentication

When you set up a REST V2 connection, you must configure the connection properties.

The following table describes the REST V2 connection properties when you use JWT bearer token authentication:

Connection property	Description

JWT Header	JWT header in JSON format.
	Sample:
	`{`
	`"alg":"RS256",`
	`"kid":"xxyyzz"`
	`}`
	You can configure `HS256` and `RS256` algorithms.

JWT Payload	JWT payload in JSON format.
	Sample:
	`{`
	`"iss":"abc",`
	`"sub":"678",`
	`"aud":"https://api.box.com/oauth2/token",`
	`"box_sub_type":"enterprise",`
	`"exp":"120",`
	`"jti":"3ee9364e"`
	`}`
	The expiry time represented as **exp** is the relative time in seconds. The expiry time is calculated in the UTC format from the token issuer time (`iat`).
	When `iat` is defined in the payload and the expiry time is reached, mappings and Generate Access Token will fail. To generate a new access token, you must provide a valid `iat` in the payload.
	If `iat` is not defined in the payload, the expiry time is calculated from the current timestamp.
	To pass the expiry time as a string value, enclose the value with double quotes. For example:
	`"exp":"120",`
	To pass the expiry time as an integer value, do not enclose the value with double quotes. For example:
	`"exp":120,`

Authorization Server	Access token URL configured in your application.

Authorization Advanced Properties	Additional parameters to use with the access token URL. Parameters must be defined in the JSON format. For example:
	`[\{"Name":"client_id","Value":"abc"},\`
	`{"Name":"client_secret","Value":"abc"}]`

TrustStore File Path	The absolute path of the truststore file that contains the TLS certificate to establish a one-way or two-way secure connection with the REST API. Specify a directory path that is available on each Secure Agent machine in the runtime environment. You can also configure the truststore file name and password as a JVM option or import the certificate to the following directory: `<Secure Agent installation directory\jre\lib\security\cacerts.`

Connection property	Description
TrustStore Password	The password for the truststore file that contains the SSL certificate. You can also configure the truststore password as a JVM option.
KeyStore File Path	Mandatory. The absolute path of the keystore file that contains the keys and certificates required to establish a two-way secure communication with the REST API. Specify a directory path that is available on each Secure Agent machine in the runtime environment. The keystore file must be in the JKS format. You can also configure the keystore file name and location as a JVM option or import the certificate to any directory.
KeyStore Password	Mandatory. The password for the keystore file required for secure communication. You can also configure the keystore password as a JVM option.
Private Key Alias	Mandatory. Alias name of the private key used to sign the JWT payload.
Private Key Password	Mandatory. The password for the keystore file required for secure communication. **Note:** The private key password must be same as the keystore password.
Access Token	Enter the access token value or click **Generate Access Token** to populate the access token value.
Swagger File Path	The absolute path along with the file name or the hosted URL of the swagger specification file. If you are providing the absolute path of the swagger specification file, the swagger specification file must be located on the machine that hosts the Secure Agent. **Note:** You can generate the swagger specification file from Administrator. Click Administrator > Swagger Files to generate a swagger specification file.
Proxy Type	Type of proxy. You can select one of the following options: - No Proxy: Bypasses the proxy server configured at the agent or the connection level. - Platform Proxy: Proxy configured at the agent level is considered. - Custom Proxy: Proxy configured at the connection level is considered.
Proxy Configuration	The proxy configuration format: `<host>:<port>` **Note:** Proxy server with authentication is not supported.
Advanced Fields	Enter the arguments that the Secure Agent uses when connecting to a REST endpoint. You can specify the following arguments, each separated by a semicolon (`;`): `ConnectionTimeout`: the wait time in milliseconds to get a response from a REST endpoint. The connection ends after the connection timeout is over. Default is the timeout defined in the endpoint API. **Note:** If you define both the REST V2 connection timeout and the endpoint API timeout, the connection ends at the shortest defined timeout. `connectiondelaytime`: the delay time in milliseconds to send a request to a REST endpoint. Default is 10000. `retryattempts`: number of times the connection is attempted. Default is 3. `qualifiedSchema`: specifies if the schema selected is qualified or unqualified. Default is false. Example: `connectiondelaytime:10000;retryattempts:5`

Note: The `HS256` algorithm support in **JWT Header** is available for preview. Preview functionality is supported for evaluation purposes but is unwarranted and is not production-ready. Informatica recommends that you use in non-production environments only. Informatica intends to include the preview functionality in an

upcoming release for production use, but might choose not to in accordance with changing market or technical circumstances. For more information, contact Informatica Global Customer Support. To use the functionality, your organization must have the appropriate licenses.

Rules and guidelines for REST V2 connections

The following verifications take place when you test a connection:

- Path of the local swagger file.
- User ID and Password fields for the Basic and Digest authentication.
- Format of the swagger file is JSON.
- Accessibility of the URL specified in the swagger file

The following table lists the results for various proxy settings use cases:

System Proxy	REST V2 Connection Attribute			Result
	No Proxy	Platform Proxy	Custom Proxy	
No	Yes	No	No	Proxy is not considered.
No	No	Yes	No	Proxy is not considered.
No	No	No	Yes	Custom proxy is considered.
Yes	Yes	No	No	Proxy is not considered.
Yes	No	Yes	No	Platform proxy is considered.
Yes	No	No	Yes	Custom proxy is considered.

CHAPTER 3

REST V2 operations

This chapter includes the following topics:

REST V2 operations overview

Create a mapping in the **Mapping Designer** to read or write data to the web service application.

When you create a mapping, you must select a REST V2 connection and an operation. The Secure Agent connects to the web service application to access, transform, or deliver data. If you want to write data to a relational target, the Secure Agent converts the hierarchical data fetched from the web service application to relational data. If you want to read data from a relational source and write data to a REST V2 target, the Secure Agent converts the relational data fetched from the source to hierarchical data.

REST V2 Connector supports all the HTTP 300 series status codes that indicate URL redirection in Source, Target, and Midstream transformations.

You can use REST V2 Connector to parse responses that are compressed using GZIP format only for GET method. You can also use REST V2 Connector to perform paging in Source and Midstream transformations.

REST V2 source operations

Create a Source transformation in the Mapping Designer to read data from the web service application.

When you select a REST V2 connection for a Source transformation in a mapping to read data from a web service application, specify Operations on the **Source** tab of the Source transformation. The Secure Agent displays operation IDs specified in the swagger specification.

Select an operation ID, configure the request message from the sample template provided in the request message editor, and configure the advanced properties for the operation. If you want to write to a relational target, define a relational structure for the source data by mapping the incoming fields that is in hierarchical format to the output fields in relational format. When you run the mapping, the Secure Agent retrieves data for the specified operation from the web service application.

Configuring a request using Request Message Editor

When you create a Source transformation, configure an XML request message for the operation that you want to perform in the web service application.

Use the **Request Message Editor** to create a request message. The request message is in XML format. You can use the sample request message for the operation and then customize the request message to specify the data that you want to enter into the data flow.

To customize your request, copy the request message from the sample template to the **Request Message Editor** pane where you can edit the XML message and add the attributes for the request. Remove unnecessary and empty tags from the request message to avoid operation failure.

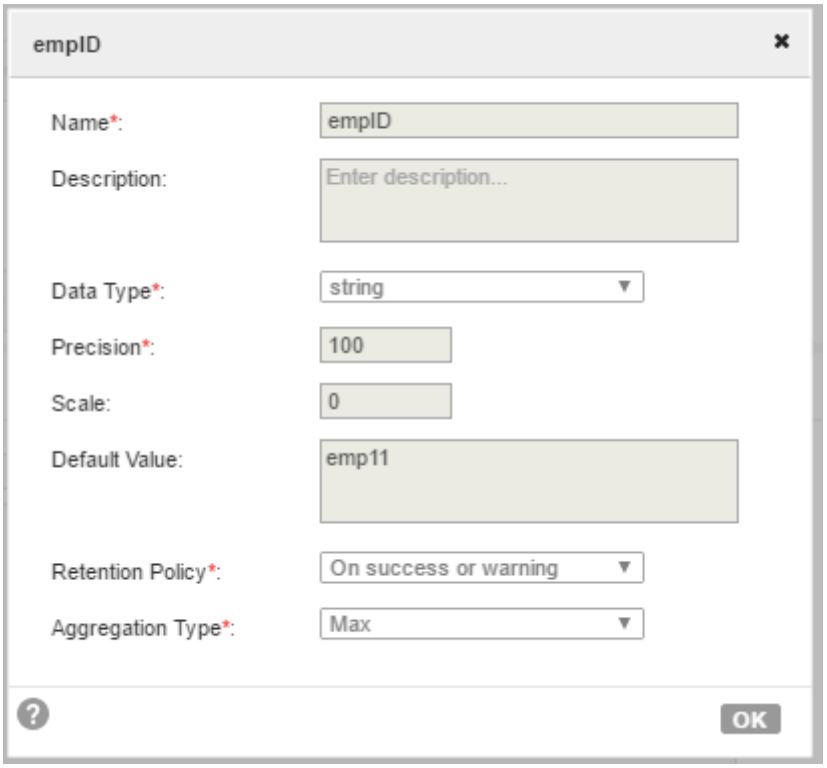
Header fields, path parameters, and query parameters appear as fields in source, target, midstream transformations

Parameterizing the input values in a request message

You can use in-out parameters to represent input values in a request XML.

Configure the in-out parameters in the Mapping Designer. From the **Mapping Design** page, you open the parameters panel and configure an in-out parameter.

The following image shows a configured `empID` in-out parameter value:



After you configure a parameter, use the parameter name in the following format, $$Name, in a request message. The XML uses the values of fields from the parameterized object.

For example, you want to use parameterized `empID` in an XML Request for a `CouchDB_INPUT` operation.

The following sample request shows the parameterized values that you can specify in an XML request:

```
<!--1 or more repetitions:-->
<proc:CouchDB_INPUT xmlns:proc="http://xml.schemas/infa/procedure/">
  <!--Optional:-->
  <CouchDB>
    <!--1 or more repetitions:-->
    <emp>
      $$empID
    </emp>
  </CouchDB>
</proc:CouchDB_INPUT>
```

Configuring values for in-out parameters in a mapping

You can use an in-out parameter that holds a variable value that can change each time a task runs to manage incremental data loading.

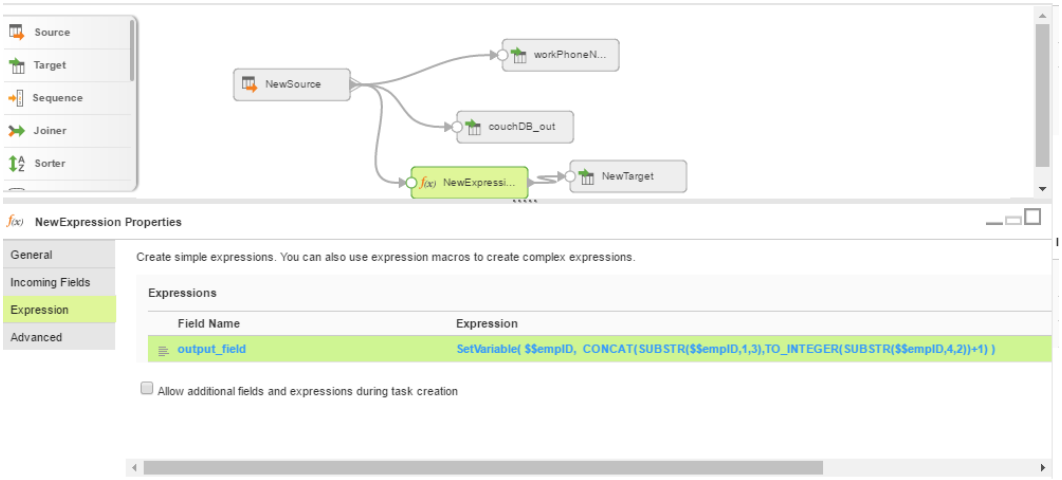
Add the in-out parameterized field in complex expressions in a mapping if you want to update the parameters when each task runs. The in-out parameter acts like a placeholder for a value that stores a task stage. Data Integration evaluates the parameter at runtime based on the specified configuration.

For example, you can use the following parameterized values in an Expression transformation:

```
SetVariable( $$empID, CONCAT(SUBSTR($$empID,1,3),TO_INTEGER(SUBSTR($$empID,4,2))+1) )
```

In the example, SetVariable function sets the parameter value each time the mapping task runs. You set a default value for the $$empID parameter. You want to update the value of $$empID by one every time the task runs.

The following image shows the logic used in expression transformation to assign value to in-out parameter in the Mapping Designer:



When the task runs, the Secure Agent updates the in-out parameters based on the specified logic in the expression.

Field mapping in a source transformation

The response message format follows the service response definition in the swagger specification file. You can map response fields from a hierarchical to a relational structure of output groups and fields.

After you configure an operation for a source and specify the request message, create the relational format from the hierarchical data to include groups and fields that you want in the output.

To do this, select the elements in the response structure that you want to include as output fields. The Secure Agent converts the XML response in the hierarchical structure to relational groups at run time.

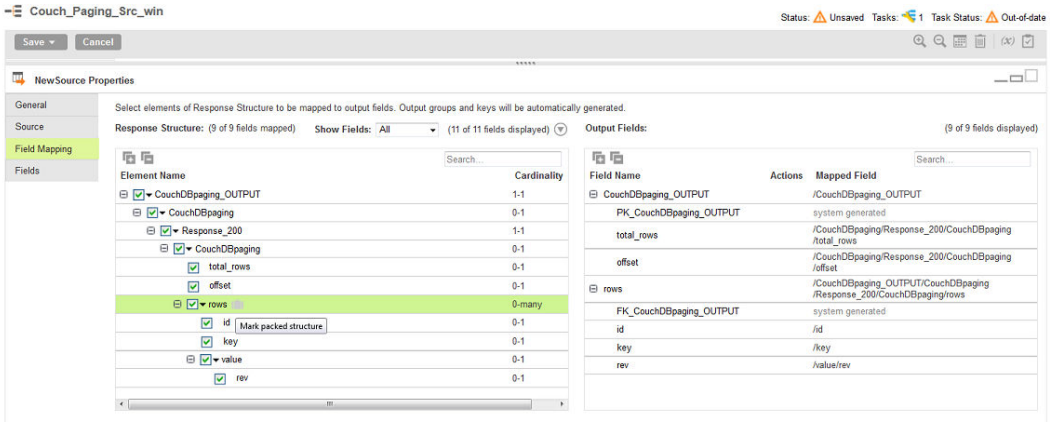
Creating packed fields

If you do not want to parse hierarchical elements, you can pack the elements into one field. You can also pack array elements when you write to the target.

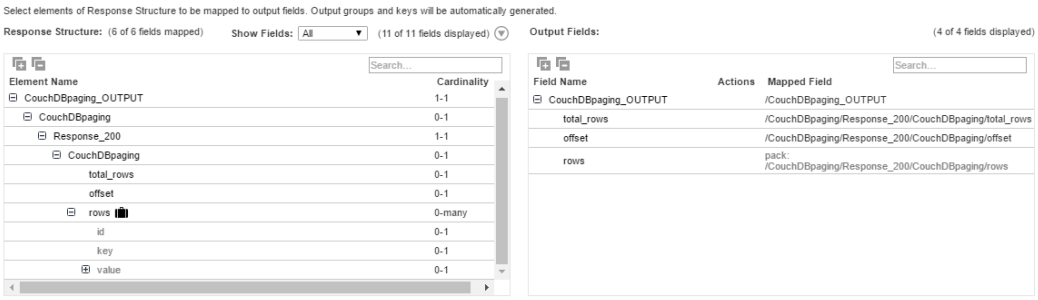
A pack icon appears next to elements on the **Field Mapping** tab. When you select the pack icon that appears next to the element, the agent packs the element and its child elements into a single XML string during runtime. You cannot select fields that are child elements of packed fields in field mapping.

You can unpack fields by clicking on the pack icon.

To pack the fields, click on the pack icon that appears next to the rows element.



The following image shows the packed rows element:



The following sample shows the output of the Rows field in the target file after you run the task:

```
<infa_packed>
<rows>
<id>52835a523b9b6816ec81e9585b09c987</id>
<key>52835a523b9b6816ec81e9585b09c987</key>
<value>
<rev>1-c7fe1b7b899a9ce50319a47d675af735</rev>
</value>
</rows>
</infa_packed>
```

REST V2 midstream operations

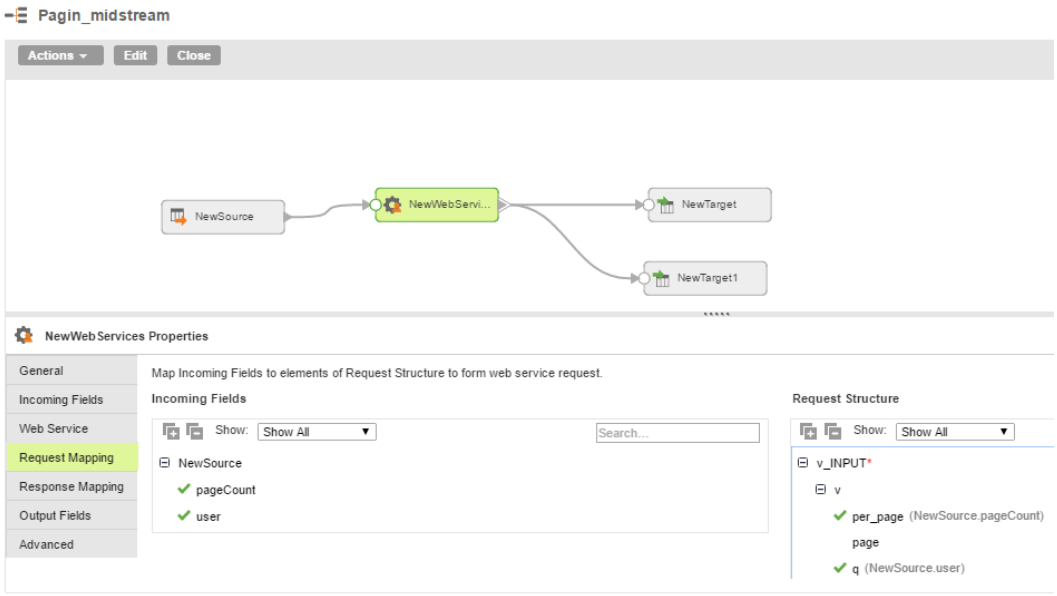
Create a REST V2 midstream transformation that interacts with the web service application to perform operations on hierarchical data.

When you use REST V2 Connector midstream, you create a business service, associate a REST V2 connection, and the required REST read or write operation for the business service. You can then use the business service to add operations to the Web Services transformation in the Mapping Designer.

Note: When you configure a business service, the values that appear in the **Operations** list are based on the operation IDs listed in the swagger specification file. If the operations IDs listed in the swagger specification file do not appear in the **Operations** list, contact Informatica Global Support.

A Web Services transformation connects to the web service application as a web service client to access, transform, or deliver data. Use the Web Services transformation in the Mapping Designer to construct a web service request and to parse the web service response.

The following image shows a REST V2 midstream transformation in a mapping:



On the Response Mapping tab of the Web Service transformation, the request generates a response structure for the specified REST operation from which you can select the elements that you require in the output. The output groups and keys generate automatically.

The response structure displays the fault output group by default. The fault group contains the request XML, error code, and error message. The fault group is displayed for the new business services. To map the fault group for the business services created in the previous version of connector, create new business services and import them again.

REST V2 target operations

Create a Target transformation in the Mapping Designer to write data to a web service application. When you select a REST V2 connection for a Target transformation in a mapping, the Secure Agent displays operation IDs specified in the swagger specification you configure in the connection. You must select an operation

based on the object configured in the connection. When you want to write data that is in relational format to the web service application, the Secure Agent converts the relational data to a hierarchical format before writing to web service application.

When you configure a REST V2 target, you can select **Operations** on the **Target** tab to display the list of valid operations. The Secure Agent creates target fields based on the request message structure of the operation you select on the **Target** tab.

When you write data to a web service application, the Secure Agent includes all the mapped incoming fields from the source. You can add multiple input sources to write data to a web service application target object and define the primary and foreign key relationships between the input sources before you run the mapping. Each input source appears as separate group in the Field Mapping tab in the target transformation. You can define key relationships in the Field Mapping tab.

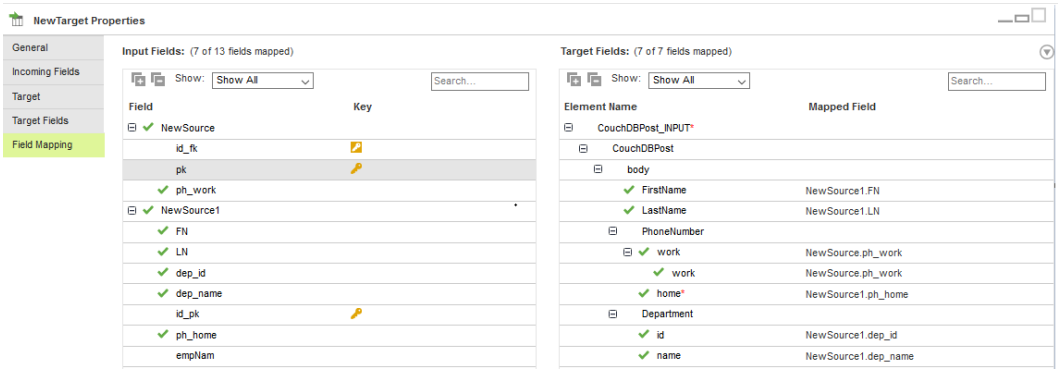
Field Mapping in a target transformation

You can map fields that are in a relational structure to the hierarchical structure used by the web service application.

On the **Field Mapping** tab for the Target transformation, the fields in the **Target Fields** appear in hierarchical format. The target fields are determined by the request message structure of the operation you select for the Target transformation.

Each source object displays as a group in the **Input Fields** area. You can select fields in the Input Fields area to map the fields to the request. If the input fields include multiple input groups, map the groups to the corresponding nodes in the request. Be sure to map all of the fields that you require in the request.

The following image shows an example of mapped input fields from the source file with the REST V2 target to update employee details:



Configuring a request that contains array elements

You can construct a request for a REST V2 target and REST V2 midstream transformation that contains array elements using one of the following methods:

Creating a request using multiple source groups

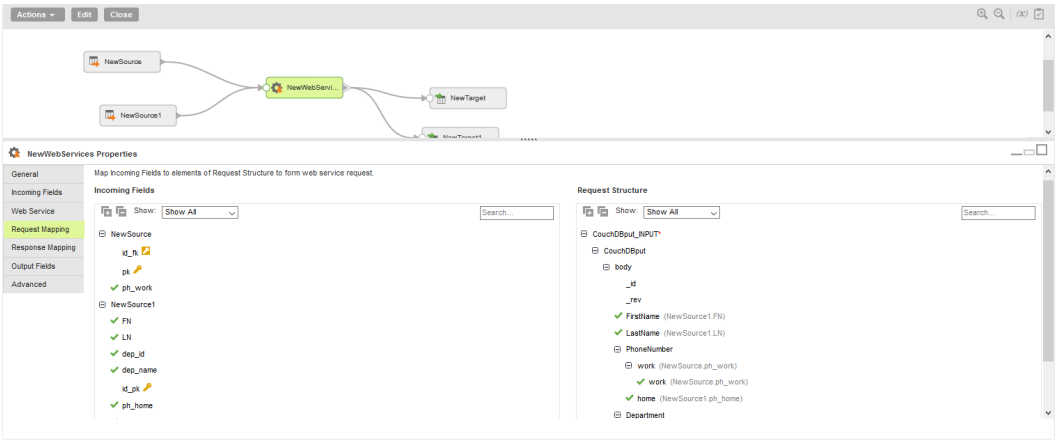
If the request message contains an element, for example, phone number of type work of an employee, that occurs multiple times, the input fields for this element must come from different source groups.

To create a request for relational sources, link the source group in the input fields to its parent source group by using the foreign key.

To decide the parent source group for multi-occurring elements, you must perform the following steps:

- The parent group must be the immediate multi-occurring parent element.
- In the absence of a multi-occurring parent element, map the multi-occurring element to the source group that is mapped to the root element.

For example, the following image shows the mapping of the multi-occurring element `work` from the relational data source to the REST V2 target:

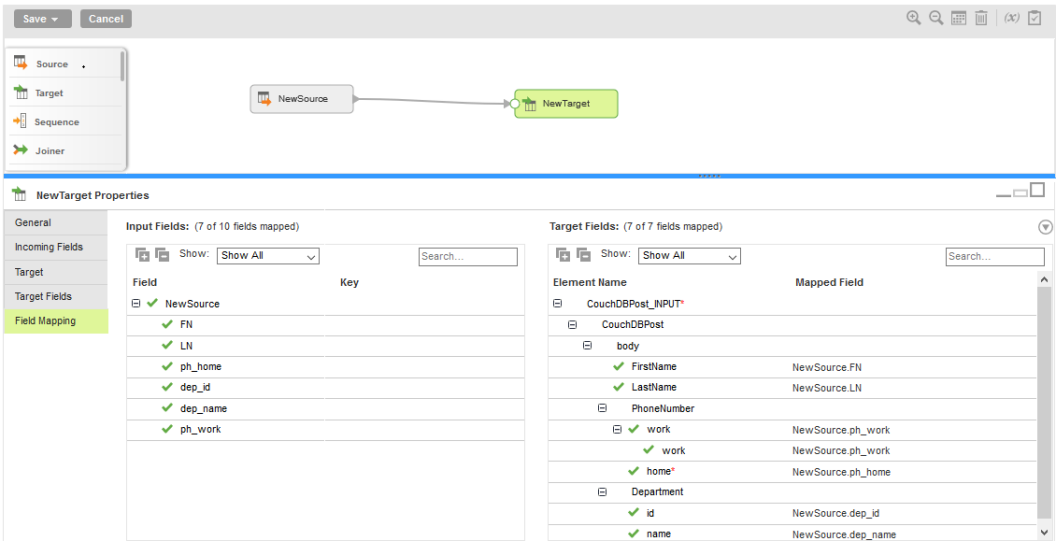


Creating a request using a single source group

To create a request using a denormalized source for a midstream transformation or REST V2 target, perform the following steps:

1. Include all the fields in a single source file. Ensure that the multi-occuring elements are the last fields in the denormalized source file.
2. Click Administrator > Runtime Environments and select an agent.
3. Add a new property under **Custom Configuration Details**.
4. Select **Service** as Data Integration Server and **Type** as Tomcat.
5. Specify the name of the property as `ALLOW_DUPLICATE_VALUES`.
6. Set the value to `true`.
7. Click **Done** to save the changes.
8. Map the corresponding source to target fields.

For example, the following image shows the mapping of the field work to an array element of REST V2 target:



Uploading a file to a REST endpoint URL

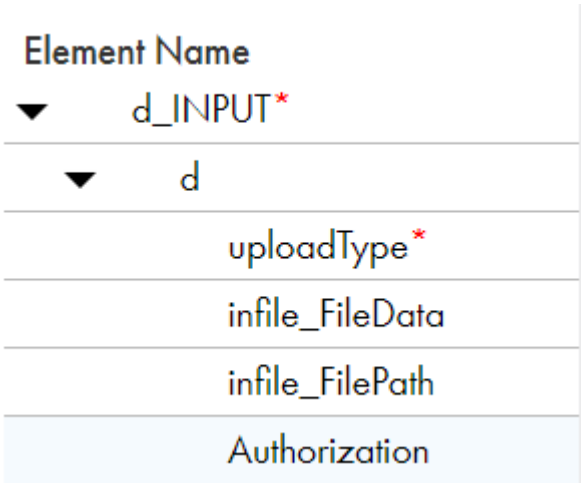
You can upload a file to a REST endpoint URL as a part of the REST API call.

To pass a file as an input to REST V2 connector, you must set the content-type to either `multipart/form-data` or `binary` in the request. When the swagger definition has Input parameter of type formData, the file boundary is added to the content of the uploaded file to indicate the start and end of the file. When the swagger definition has Input parameter of type `binary`, the content is generated without having the file boundary.

You must have the `formData` parameter of type `file` defined as one of input parameters in swagger. For example,

```
{ "name":"file",

"in":"formData",

"description":"file to upload",

"required":false,

"type":"file" }
```

The following image shows the sample REST V2 object hierarchy:



Use one of the following methods to pass a file as an input:

- For the `xxx_FilePath` input field, specify the complete file path as value in the source. Used for any file formats.
- For the `xxx_FileData` input field, pass the Base64 encoded value of the file in the source. Used for file formats, such as `.pdf`, `.jpg`, `.xls`, and `.doc`. The length of the Base64 encoded value must not exceed 65535 characters.
- For the `xxx_FileData` input field, pass the file value as string in the source. Used for plain text file formats, such as `.txt`, `.JSON`, and `.xml`. The file size must not exceed 65535 characters.

Parsing response headers

You can parse the header content in a response sent by a REST API endpoint.

Use the `headers` tag in the swagger file to parse the header content. By using the `Set-Cookie` tag, you can also parse the cookie content as part of the `header` tag. REST V2 Connector parses fields defined in the swagger file. Any extra fields coming from the web service for header or cookies appear under `OtherResponseHeaders` or `OtherCookies` fields respectively.

Defining Response Headers

```
"responses" : {
....
"headers": {
"places": {
"type": "string",
"description": "calls per hour allowed by the user"
},
"sessionId": {
"type": "integer",
"description": "date in UTC when token expires"
```

```
},
"Set-Cookie/activeSession": {
"type": "string",
"description": "cookies sample"
},
"Set-Cookie/PhoneId": {
"type": "integer",
"description": "cookies sample"
}
}
```

Note: The `Set-Cookie` tag is case-sensitive.

Parsing security definitions

You can define security definitions when you perform an operation on a REST API endpoint.

Define the `securityDefinitions` input parameter in the swagger file and call the security definitions in an operation. The `securityDefinitions` parameter supports only the basic and apiKey security definitions types.

Defining securityDefinitions

```
...
"security": [{"type": [],"id": [],"basic_key": []}]
}
}
},
"securityDefinitions": {
"type": {
"type": "apiKey",
"name": "type",
"in": "query",
"default":"json"
},
"id": {
"type": "apiKey",
"name": "id",
"in": "query"
},
```

```
"basic_key": {
"type": "basic",
"name": "basicAuth",
"in": "header"
}
}
```

If you want to connect to a web service that uses OAUTH version 2.0, provide the values of authentication details as custom header fields or query parameters in a request message.

Rules and guidelines for REST V2 operations

Consider the following rules and guidelines for REST V2 Operations:

- Elements related to header fields, path parameters, and query parameters appears as input fields in Source, Target, and Midstream transformation.
- REST V2 Connector does not support a proxy server on which authentication is enabled.
- REST V2 Connector supports only Integer and Decimal data types.
- REST V2 Connector supports packed field in Source transformation only.
- REST V2 Connector overrides the Content-Type and Accept headers.
- REST V2 Connector does not support extended ASCII characters and I18N characters in a field name for JSON payload.
- REST V2 Connector does not support special characters in the field name for query parameters and operation names.
- REST V2 Connector does not support special characters in the array type fields.
- You must not create in-out parameters for advanced properties, such as Page Parameter, Start Page, End Page, and Override URL.
- If an endpoint API path contains spaces, you must define the path as path parameter.
- If the swagger URL you specify is not accessible or the proxy server is not accessible, mappings fail and one of the following errors is logged in the session log. The behavior is same for the source, midstream, and target mappings.

  - `[ERROR] com.informatica.sdk.helper.common.ApiException: java.net.UnknownHostException: inv28pers108Communication exception, Proxy settings might be incorrect.`
  - `[ERROR] com.informatica.sdk.helper.common.ApiException: java.net.ConnectException: Connection refused: connectCommunication exception, Proxy settings might be incorrect.`
- You can define a page parameter on the request input field if the payload is of type JSON. The page parameter gets resolved only when the JSON request payload is an unnamed object. For example:

  ```
  "body" : {
  "a" : "ty",
  "b" : 0
  }
  ```

The page parameter does not get resolved in the following scenarios:

- When the JSON request payload is a named object.
- The parameter defined is a nested input element.
- The parameter defined is an array element.
- When the payload is of type XML.

- You can define default values for request input parameters except for the `body` and `securityDefinitions` parameters. For example:

```
"parameters": {
"keyReference": {
"name": "keyReference",
"in": "header",
"required": false,
"type": "integer",
"default": 123
}
}
```

  Note: Default values are applicable to integer, string, boolean, and array of primitive types fields.
- When you use the REST V2 Connector as source, the default values defined in the swagger file are not honored if you pass empty values for input parameters in the request message editor. If you do not provide values in request message fields and want the fields to pick up default values defined in the swagger definition, you must remove the fields from the request message editor.
- If the default value for an input parameter is defined as blank, the blank value is treated as NULL unless you set the value explicitly in a mapping.
- You cannot define XML attributes using inline elements in the swagger file. Define XML attributes as separate elements in the swagger file and refer the XML attributes to extract data.

CHAPTER 4

Mappings and mapping tasks with REST V2 Connector

This chapter includes the following topics:

Mappings and mapping tasks overview

When you create a new mapping in this release, the request message editor shows the sample request in the JSON format for the JSON data. Previously, the sample request for the JSON data was shown in the xml format. This changed behavior occurs because in this release, REST V2 Connector uses the JSON parser shipped with Data Integration. If you do not change the mappings created in previous release, the mappings continue to use the old parsing technique and run successfully.

If you update the business service, source, or target either by changing the connection or the operation in the mappings created in the previous release, the Informatica JSON parser is enabled. The updated mapping might throw a null pointer exception during metadata fetch when the schema related to the operation does not have a write permission.

REST source transformation in mappings

When you configure a Source transformation, select the REST V2 connection and choose an operation to represent a web service source.

You can select an operation for the source through the connection. Configure the request message using the request message template. You can parameterize the input values in the request XML. Configure the paging attributes.

You can view the response structure in the field mapping. When you map the elements from the response structure to the output fields, the Secure Agent creates the output groups, along with the primary and foreign keys for the field names. When you deploy the mapping in a mapping task and run the task, the Secure Agent reads the data from web service.

Advanced source properties

In a mapping, you can configure a source to represent a web service application source. For the REST source connections used in a mapping, you can configure advanced properties in the **Source** tab in the Mapping Designer.

The following table describes the advanced properties that you can configure in a source:

Property	Description
Paging Type	Specify one of the following values: **Page**. Enables paging support for REST V2 Connector and considers the values of Page Parameter, Start Page, End Page, and End of Response Expression properties. **None**. Ignores the values of Page Parameter, Start Page, End Page, and End of Response Expression properties.
Page Parameter	The name of the parameter that you want to use for the paging operation. You can use a query parameter or a path parameter. The parameter must be of the integer type and from the request message.
Start Page	The page number that indicates the first page in the range, on which you want to perform the paging operation.
End Page	The page number that indicates the last page in the range, on which you want to perform the paging operation. The default is 10000. Paging stops when the End Page is reached or the End of Response Expression is met.
Page Increment Factor	An integer to increment the Page Parameter. Page Increment Factor must be same as the number of records being fetched per request. **Note:** If the Page Increment Parameter option is not same as the number of records being fetched per request, you might have missing or duplicate records between two calls.
End of Response Expression	Specify an expression or a string to control paging. You can observe one of the following behaviors: - If you specify a string or an expression, the paging stops when the value matches with the page response. It does not parse the page that has matching end of response. - If you do not specify a string or an expression, the paging stops on reaching a page that has an empty or a Null response. If an empty or a Null response is never reached, the paging stops at the default end page. - If you specify both, End Page and End of Response Expression, the paging stops on whichever condition is met first. - If you do not specify End Page and End of Response Expression, the paging stops on reaching an empty or a Null response. If an empty or a Null response is never reached, the paging stops at the default end page.
Override URL	Overrides the URL specified in the swagger specification. The override URL cannot have query parameters. When a path parameter is included in the Override URL, enclose the path parameter with curly brackets {}. For example: URL specified in the swagger specification: `http://invr28pers102:13080/sample/day/20170505?a:b` If you define 20170505 as a path variable with the path variable name as `path1`, the Override URL will be as follows: `http://invr28pers102:13080/sample/day/{path1}`

Property	Description
Tracing Level	Amount of detail that appears in the log for the source. Use the following tracing levels: - Terse - Normal - Verbose Initialization - Verbose Default is normal.
Cache Size for Web Service Response (KB)	Memory available for the web service response. If the web service response contains many rows or columns, you might want to increase the cache size. Default is 1024 KB.

Sample End of Response Expression

Use End of Response Expression for paging. The following snippet shows a sample response for a page :

```
{
"code": "SUCCESS",
"validationResult": [],
"systemErrors": [],
"patientResponseData": [],
"count": 0,
"message": "Unable to retrieve the implant details"
}
```

For End of Response Expression, you can use the string, Unable to retrieve the implant details. If you want to match multiple conditions in the response page, you can use the following expression:

```
(.*)"patientResponseData": [](.*)Unable to retrieve the implant details
```

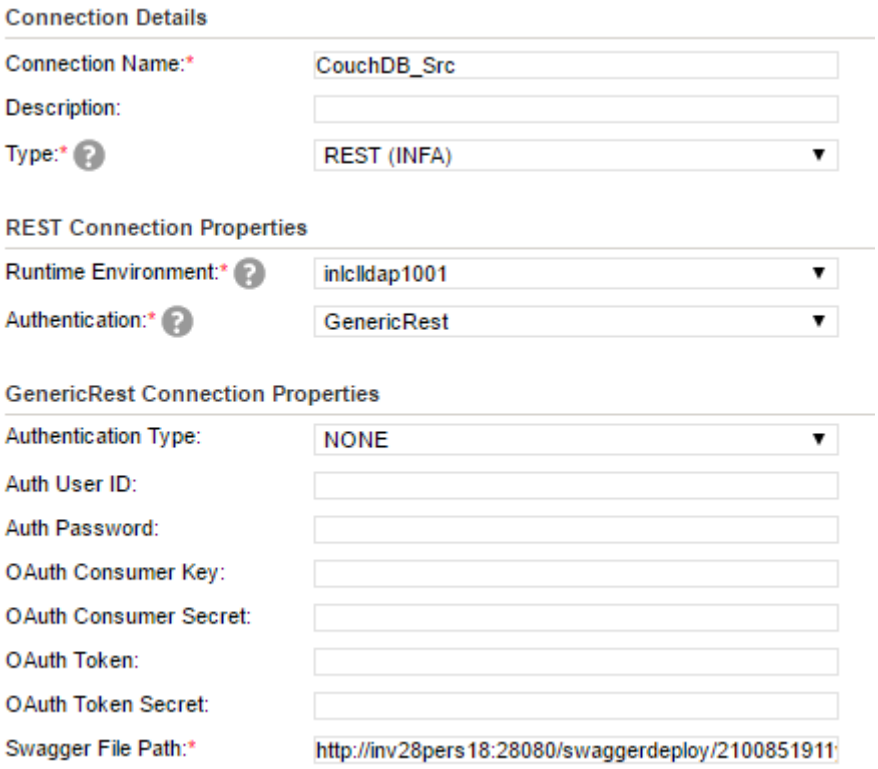
The above expression ensures that the paging will stop when both "patientResponseData": [] and Unable to retrieve the implant details are matched in the page response.

REST source mapping example

You are a human resources administrator and you want to extract contact information, such as first name, last name, email, and phone number of employees from Apache CouchDB to a flat file.

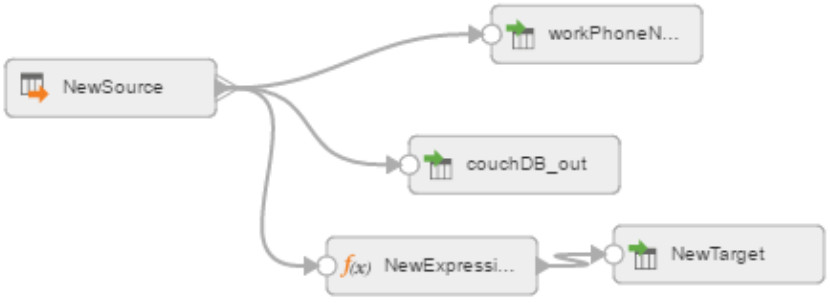
1. Create a REST connection. Verify that you specify the absolute path or the hosted URL of the swagger specification file, and the authentication method in the connection properties.

   The following image shows the configured REST connection:



2. Create a flat file connection to write data to the flat file.
3. Create a REST mapping.

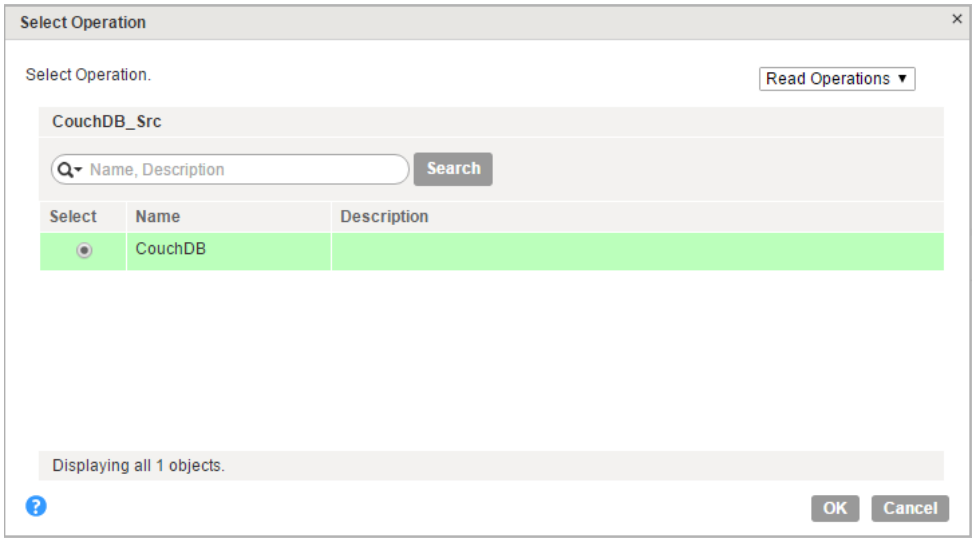
   The following image shows the REST mapping:



4. Add a Source transformation. Specify a name and description in the general properties.
5. On the **Source** tab, perform the following steps:
   a. In the **Connection** field, select the configured REST connection to connect to the Couch database.

b. In the Operation field, select CouchDB as the operation.

The following image shows the CouchDB operations:



c. In the **Request Options** section, configure the request message in the following XML format, specify the attributes in the message, and validate the message:

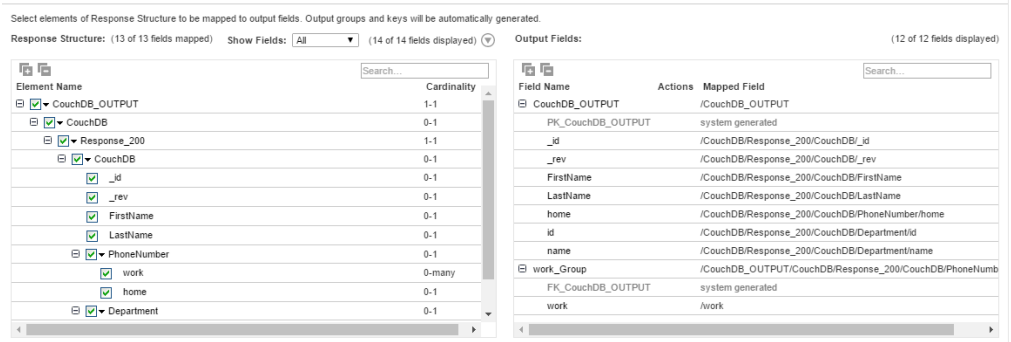
```
<!--1 or more repetitions:-->
<proc:CouchDB_INPUT xmlns:proc="http://xml.schemas/infa/procedure/">
  <!--Optional:-->
  <CouchDB>
    <!--1 or more repetitions:-->
    <emp>
      $$empID
    </emp>
  </CouchDB>
</proc:CouchDB_INPUT>
```

The request message fetches the details of an employee based on the specified employee ID.

d. In the **Advanced Properties** section, set the tracing level to Normal, and use the default cache size of 1024 KB.

6. On the Field Mapping tab, select the elements, such as `_id`, `FirstName`, and `LastName` in the response structure that you want to map to the output fields.

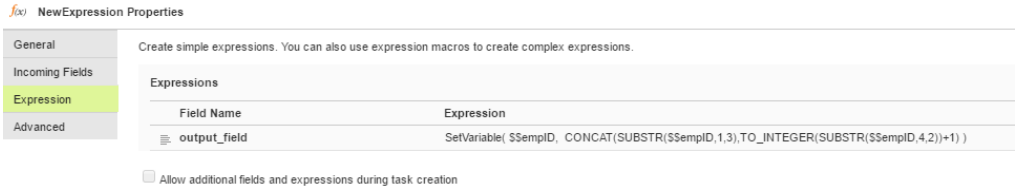
The following image shows the response structure on the left pane in a hierarchical format and the output groups on the right pane in a relational format:



The Secure Agent creates two output groups, workPhoneNumber and couchDB_out, which results in two relational output files. Primary and foreign keys are auto-generated.

7. Add three Target transformations and specify the target objects for each of the transformations. Perform the following steps:
   a. Select a flat file connection for each of the target transformations to write data to the flat files.
   b. Create target objects workPhoneNumber.csv, couchDB_out.csv, and NewTarget.csv files to write data to the target.
   c. Select the output group from CouchDB that you want to link to the target objects.
8. Add an Expression transformation, and include a `SetVariable` function and the parameterized value `$$empID` in the expression so that the task updates the value of `empID` by one at the end of each run.

   The following image shows the configured expression that utilizes the in-out parameters:



9. Click **New** > **Tasks** > **Mapping Task**, and select the mapping for the task.
10. When you save and run the mapping task, the Secure Agent retrieves employee records from CouchDB and writes the data to the corresponding flat files.

REST midstream transformation in mappings

Before you configure a midstream transformation, you must create a business service from the **New** > **Components** tab. You must select a REST V2 connection and an operation when you create a business service.

When you configure the midstream transformation, select the business service and the operation on the **Web Service** tab.

You can map the input fields from source file to elements in request structure on the **Request Mapping** tab.

You can view the response structure in the field mapping. When you map the elements from the response structure to the output fields, the Secure Agent creates the output groups, along with the primary and foreign keys for the field names.

Advanced midstream properties

The following table describes the advanced properties that you can configure for a midstream transformation:

Property	Description
Paging Type	Specify one of the following values: **Page**. Enables paging support for REST V2 Connector and considers the values of Page Parameter, Start Page, End Page, and End of Response Expression properties. **None**. Ignores the values of Page Parameter, Start Page, End Page, and End of Response Expression properties.
Page Parameter	The name of the parameter that you want to use for the paging operation. You can use a query parameter or a path parameter. The parameter must be of the integer type and from the request message.
Start Page	The page number that indicates the first page in the range, on which you want to perform the paging operation.
End Page	The page number that indicates the last page in the range, on which you want to perform the paging operation. The default is 10000. Paging stops when the End Page is reached or the End of Response Expression is met.
Page Increment Factor	An integer to increment the Page Parameter. Page Increment Factor must be same as the number of records being fetched per request. **Note:** If the Page Increment Parameter option is not same as the number of records being fetched per request, you might have missing or duplicate records between two calls.
End of Response Expression	Specify an expression or a string to control paging. You can observe one of the following behaviors: - If you specify a string or an expression, the paging stops when the value matches with the page response. It does not parse the page that has matching end of response. - If you do not specify a string or an expression, the paging stops on reaching a page that has an empty or a Null response. If an empty or a Null response is never reached, the paging stops at the default end page. - If you specify both, End Page and End of Response Expression, the paging stops on whichever condition is met first. - If you do not specify End Page and End of Response Expression, the paging stops on reaching an empty or a Null response. If an empty or a Null response is never reached, the paging stops at the default end page.
Override URL	Overrides the URL specified in the swagger specification. The override URL cannot have query parameters. When a path parameter is included in the Override URL, enclose the path parameter with curly brackets {}. For example: URL specified in the swagger specification: `http://invr28pers102:13080/sample/day/20170505?a:b` If you define 20170505 as a path variable with the path variable name as `path1`, the Override URL will be as follows: `http://invr28pers102:13080/sample/day/{path1}`
Cache Size for Web Service Request (KB)	Default is 100 KB. If the request is more than 100 KB, you can increase the cache size.
Cache Size for Web Service Response (KB)	Memory available for the web service response. If the web service response contains many rows or columns, you might want to increase the cache size. Default is 100 KB.

Property	Description
Allow Input Flush	Not Applicable.
Transaction Commit Control	Not Applicable.

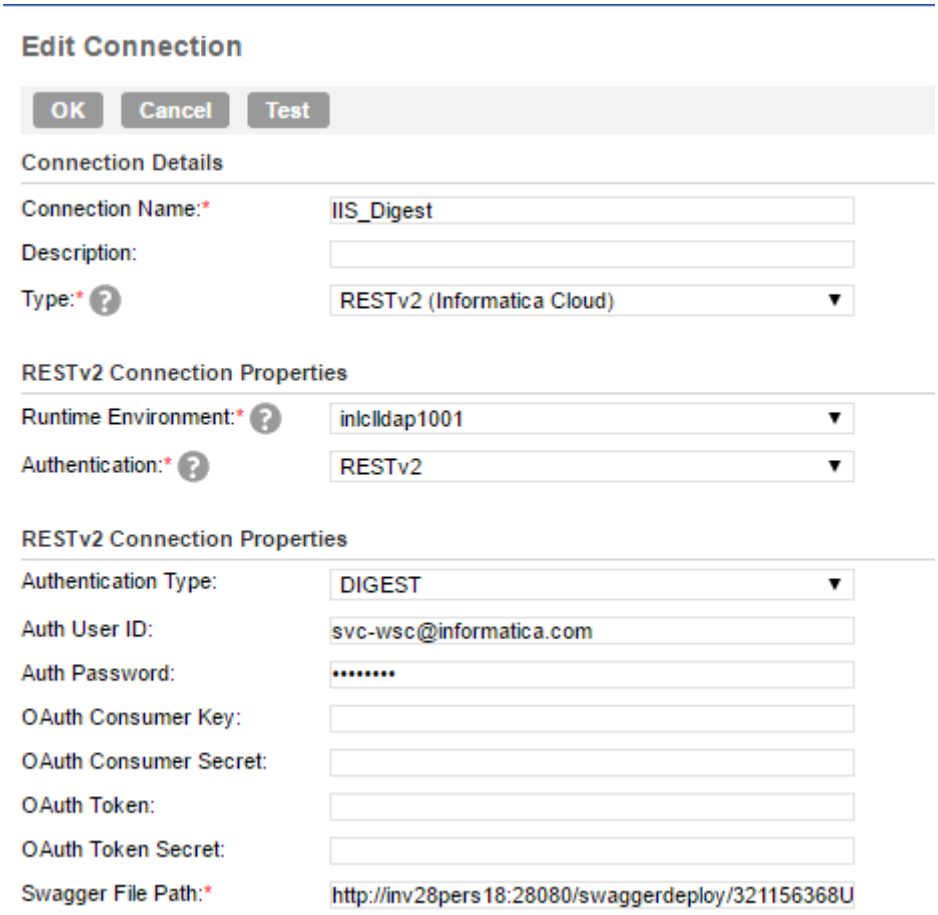
Midstream transformation mapping example

You are a human resources administrator and you want to retrieve details of an employee from Apache CouchDB.

To retrieve employee details from CouchDB and to write the employee details to a flat file, perform the following tasks:

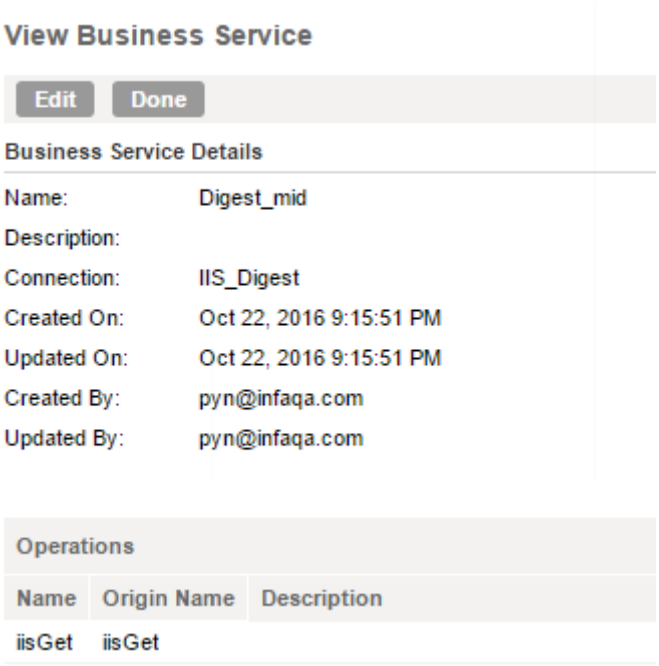
1. Create a REST V2 connection to read data from CouchDB.

   The following image shows the configured CouchDB connection:



2. Create a business service to associate the REST V2 connection and an operation.

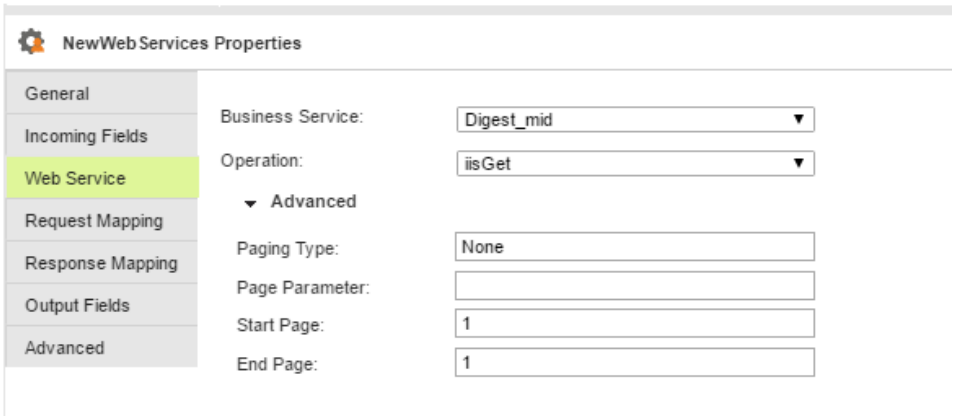
The following image shows the configured business service:



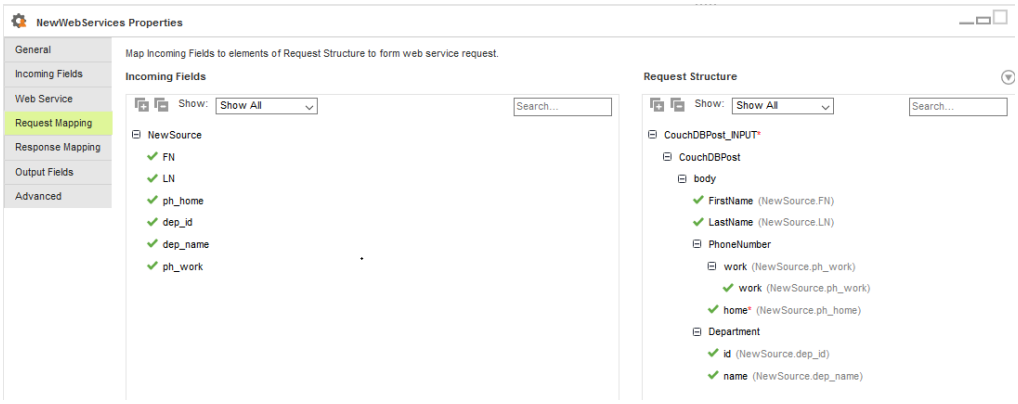
3. Create a REST V2 mapping.

4. Add a Web Services transformation. Specify a name and description in the general properties.
5. On the **Web Service** tab, select the business service and the operation that you configured. Also, specify paging parameters.

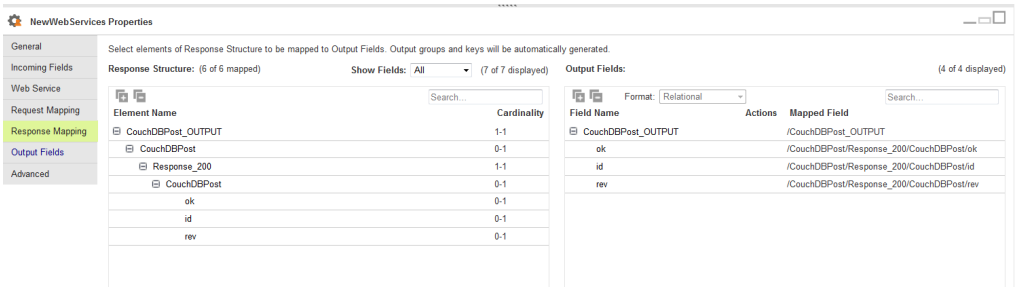
   The following image shows the configured web service:



6. On the **Request Mapping** tab, map the incoming fields from the source to the respective fields in CouchDB.



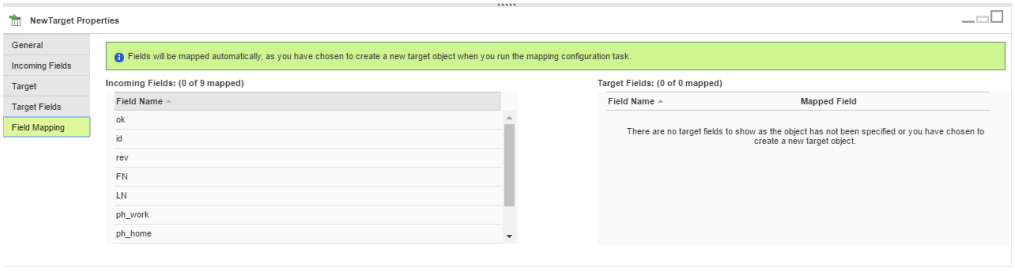
7. On the **Response Mapping** tab, select the employee details fields that you want to write to the target file on the **Response Structure**.



8. On the **Advanced** tab, specify the cache size details.

   The fields that you selected from the **Response Structure** of the Web Service transformation appear as incoming fields for the target object.

9. If required, map the incoming fields to the flat file fields.



10. Click **New** > **Tasks** > **Mapping Task**, and select the mapping for the task.
11. When you save and run the mapping task, the Secure Agent retrieves the employee record for the employee that you map on the Request Mapping tab from CouchDB and writes the data to the corresponding flat files.

REST V2 targets in mappings

When you select a REST V2 connection for a Target transformation, you can select an operation. The operation is based on the swagger specification file that you specify during the REST V2 connection configuration.

You can add multiple input groups into the REST target and define the primary and foreign key relationships between the multiple input groups before the mapping.

Advanced target properties

In a mapping, you can configure a target to represent a REST target. For REST target connections used in a mapping, you can configure advanced properties in the **Targets** page of the Mapping Task wizard.

The following table describes the advanced properties that you can configure in a Target transformation:

Property	Description
Paging Type	Specify one of the following values: **Page**. Enables paging support for REST V2 Connector and considers the values of Page Parameter, Start Page, End Page, and End of Response Expression properties. **None**. Ignores the values of Page Parameter, Start Page, End Page, and End of Response Expression properties.
Page Parameter	The name of the parameter that you want to use for the paging operation. You can use a query parameter or a path parameter. The parameter must be of the integer type and from the request message.
Start Page	The page number that indicates the first page in the range, on which you want to perform the paging operation.
End Page	The page number that indicates the last page in the range, on which you want to perform the paging operation. The default is 10000. Paging stops when the End Page is reached or the End of Response Expression is met.
Page Increment Factor	An integer to increment the Page Parameter. Page Increment Factor must be same as the number of records being fetched per request. **Note:** If the Page Increment Parameter option is not same as the number of records being fetched per request, you might have missing or duplicate records between two calls.
End of Response Expression	Specify an expression or a string to control paging. You can observe one of the following behaviors: - If you specify a string or an expression, the paging stops when the value matches with the page response. It does not parse the page that has matching end of response. - If you do not specify a string or an expression, the paging stops on reaching a page that has an empty or a Null response. If an empty or a Null response is never reached, the paging stops at the default end page. - If you specify both, End Page and End of Response Expression, the paging stops on whichever condition is met first. - If you do not specify End Page and End of Response Expression, the paging stops on reaching an empty or a Null response. If an empty or a Null response is never reached, the paging stops at the default end page.

Property	Description
Override URL	Overrides the URL specified in the swagger specification. The override URL cannot have query parameters. When a path parameter is included in the Override URL, enclose the path parameter with curly brackets {}. For example: URL specified in the swagger specification: `http://invr28pers102:13080/sample/day/20170505?a:b` If you define 20170505 as a path variable with the path variable name as `path1`, the Override URL will be as follows: `http://invr28pers102:13080/sample/day/{path1}`
Cache Size for Web Service Request (KB)	Memory available for the web service request. If the web service request contains many rows or columns, you might want to increase the cache size. Default is 1024 KB.
Transaction Commit Control	Control to commit or roll back transactions based on the set of columns that pass through the transformation. Use the transaction commit control if you have a large amount of data and you want to control how it is processed. **Note:** Does not apply when you select **Transformation Scope** as `Transaction` for real-time processing.
Transformation Scope	The method in which the Secure Agent applies the transformation logic to incoming data. Default is `All Input`. To process data from a real-time source, select the transformation scope as `Transaction`. If the source is not real-time, select the transformation scope as `All Input`. **Note:** You can connect only one effective real-time source to a real-time target.

Input settings properties

You can enable **Sorted Input** under **Input Settings**. Sorted Input indicates that input data is presorted. Default is disabled.

Enable sorted input for better performance.

Note: When **Sorted Input** is enabled and the input is not sorted, the Secure Agent does not process input and the mapping fails.

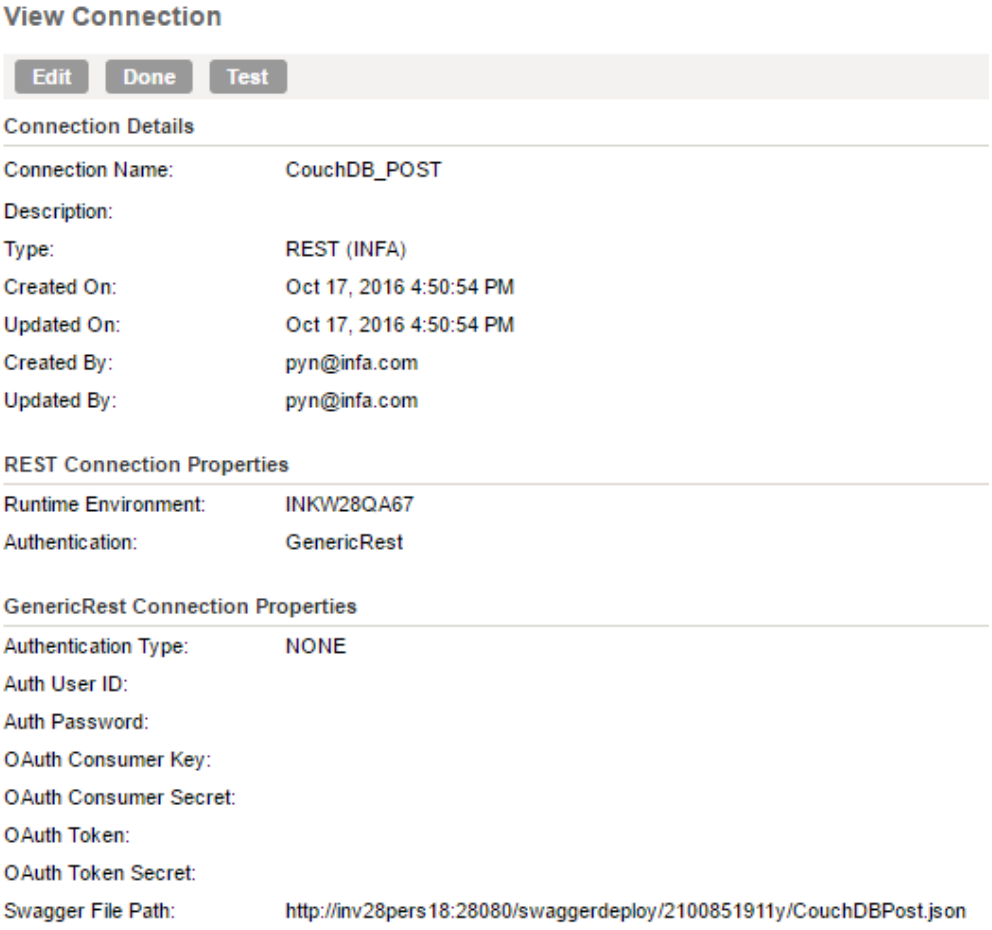
REST target mapping example

You are a human resources administrator and you want to update details of an employee in Apache CouchDB.

To update employee details in CouchDB from a flat file, perform the following tasks:

1. Create a flat file connection to read data from the flat file.
2. Create a REST connection to write data to CouchDB.

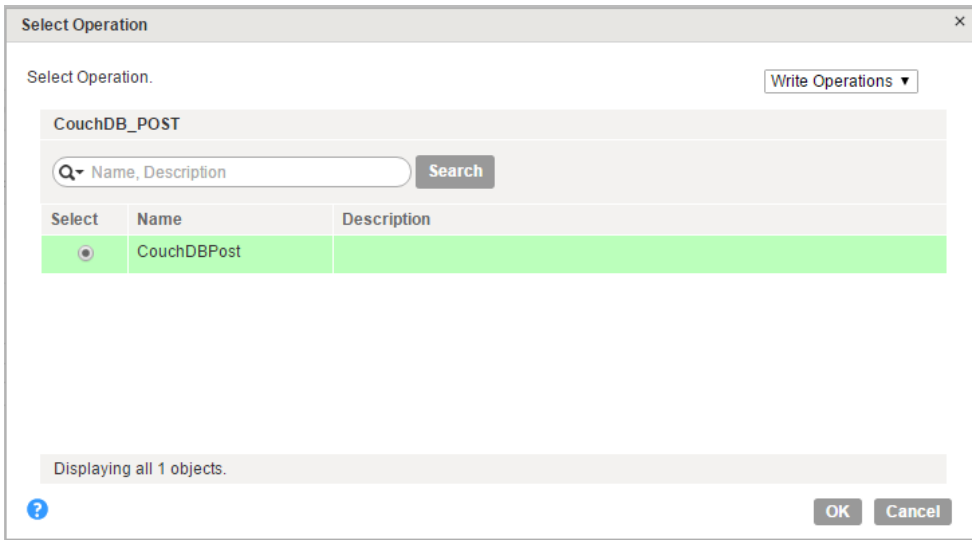
The following image shows the configured CouchDB connection:



3. Create a CouchDB mapping.

4. Add a Source transformation to include the flat file object that contains the employee details. Add the flat file connection.
5. Add a Target transformation to write the employee details to CouchDB. Perform the following tasks on the **Target** tab:
   a. In the **Connection** field, select the REST connection to connect to CouchDB.
   b. In the **Operation** field, select CouchDBPost as the operation.

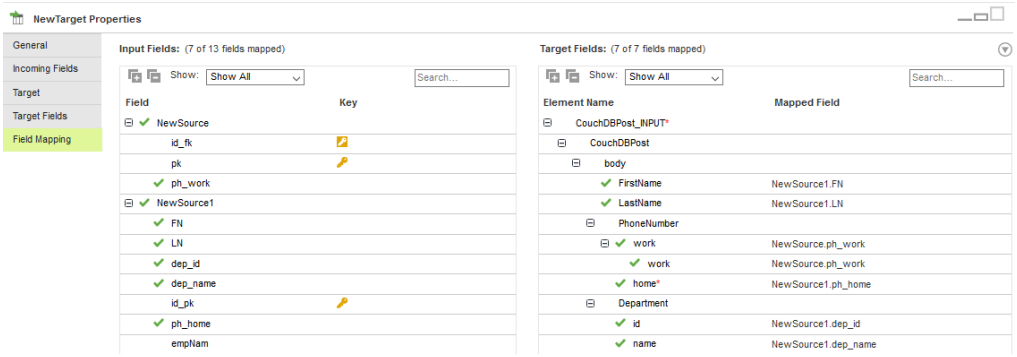
The following image shows the CouchDBPost in the list of write operations:



c. In the **Advanced Properties** section, set the cache size and the transaction commit interval.

6. On the **Field Mapping** tab, select the input elements to map to the target fields.

   The following image shows all the mapped fields between the input file and the CouchDB target:



7. When you save and run the mapping in a mapping task, the Secure Agent updates the employee details in CouchDB.

APPENDIX A

Supported swagger objects

This appendix includes the following topic:

Supported swagger objects overview

The following table lists swagger objects with field names supported by REST V2 Connector:

Object Category	Supported Objects or Fields
Swagger	swagger
	info
	host
	basepath
	schemes
	consumes
	produces
	paths
	definitions
	parameters
	responses
Paths	/{path}
Path Item	$ref
	get
	put

Object Category	Supported Objects or Fields
	post
	delete
	parameters
Operation	Summary **Note:** If operation ID value is not defined, the value of summary field is treated as operation ID.
	operationId
	consumes
	produces
	parameters
	responses
	schemes
Parameter Type	path
	query
	header
	formData
	cookie
	body
Parameter Fields	name
	In If `in` is `body`, the supported field is schema. If `in` is any value other than `body`, the supported fields are type, format, and items.
	required
	default **Note:** Not applicable to the `body` and `securityDefinitions` parameters.
Items	type
	format
	items
Response	schema
Schema	$ref

Object Category	Supported Objects or Fields
	format
	type
	items
	properties
	xml
Definitions	{name}
securityDefinitions	{name} **Note:** Basic and apiKey are the supported security definitions types.
xml	name **Note:** Name works only when used with the wrapped object.
	attribute
	prefix
	namespace
	wrapped

Rest V2 Connector does not process unsupported objects and fields, if included, in the swagger definition.

INDEX

U

W