Automic™

Let's Automate Business.

Rapid Automation

ONE Automation

Web Service SOAP Agent Guide

Version: 4.1 for the Automic Web Interface

Publication Date: 2018-03

Automic Software GmbH

Copyright

Contents

1 The RA Web Service SOAP Agent Solution

The RA Web Service SOAP Agent allows Automation Engines to be the consumer of other application's web services.

For the RA Web Service SOAP Agent solution, you will need to create one or more:

- Agents
- Connections
- Jobs

Getting the Latest Information

Documentation, release notes, and other information is often updated after software is released. The table below shows where to find the most recent information for Automic software releases.

To find the most recent:	Go to the:
Bug fixes, known issues, and workarounds	Automic Download Center
HTML5 documentation and .pdf files for documentation and release notes	Automic Hosted Documentation
Compatibility for Automic software components, versions, and sub-components	Automic Compatibility Matrix

1.1 Who Uses the RA Web Service SOAP Agent?

People in several roles use the RA Web Service SOAP Agent to accomplish different tasks. While Job titles and responsibilities differ from one company to the next, the following user groups describe the most common requirements for user roles. They are presented here to:

- Help the Automic Administrator in your organization determine the authorizations, privileges, and available clients the users in your organization should have.
- Give other users an idea of who in your organization is responsible for what.

User Groups

- **Automation Engine Administrators**
    - Has access to the system client (client 0) as well as one or more numbered clients.
    - Deploy and configure Agents and maintain their health along with the Automic infrastructure.
    - Set-up resiliency by replicating RA Web Service SOAP Agent set-up on a second machine. Generally they will only setup two RA Web Service SOAP Agent solution Agents for scalability.
    - Set-up Agents for Kerberos Forests.

  Note that Automation Engine Administrators are not the primary audience for the RA Web Service SOAP Agent. In fact, they may never use it directly. Their primary role is to set-up and configure the RA Web Service SOAP Agent solution Agents. They likely will not be the ones to create RA Web Service SOAP Agent Connection objects.

- **Security Administrators**
    - Has access to one or more numbered clients.
    - Set-up the transport security for the Agent. This includes SOCKS settings and SSL Certificates.
    - Define Connection objects with the corresponding security information.
- **Line of Business and IT Infrastructure Developers**
    - Has access to one or more numbered clients.
    - Create individual RA Web Service SOAP Agent Job and Workflow objects.
    - Create Connection objects (in some cases, possibly only in a test environment).
- **Quality Assurance Engineers**
    - Has access to one or more numbered clients.
    - Run RA Web Service SOAP Agent Jobs and Workflow for test automation and release automation.
- **Line of Business Operators**
    - Has access to one or more numbered clients.
    - Monitor and troubleshoot Workflows that use RA Web Service SOAP Agent tasks.
    - View, but not edit the Connection objects that are available to each Agent.

  Note that Line of Business Operators are likely to use operational run books that call web service for things such as escalation procedures and remediation routines. They require good documentation. They will make calls to the Amazon Cloud, VMWare's vCloud partners, and other systems to find out system status and health information.

- **Cloud Administrators**

- Has access to one or more numbered clients.
- Create and run RA Web Service SOAP Agent Jobs and Workflow to manage their virtual data centers, virtual machines, logical unit numbers, storage, etc.

1.2 About Rapid Automation Agents

Agents are the objects that establish the connection between the Automation Engine and the application or operating system where Jobs are processed. Agents start, monitor and report the current status of the Jobs.

The term Rapid Automation refers to a generic technology that is able to include and process various automation solutions in Automation Engine. The solutions are realized through Rapid Automation (RA) solutions. RA Agents make the functions of an RA solution accessible.

After loading an RA **.jar** file into the Automation Engine, you will be able to define RA Agent-specific connection, Agent, and Job objects with unique screens. The topics in this guide describe how to use these RA screens, not how to define connection, Agent, and Job objects. For information on creating these objects, see your Automation Engine or documentation.

Using Object Variables in Fields

You can use object variables in the format **&<variable name>** in most fields on Rapid Automation panels. The values of these object variables will be replaced at run time.

It is recommended that you terminate object variable names with a # character.

1.3 Using WebHelp at docs.automic.com

The format of the documentation at docs.automic.com requires a Web browser. There you will find the most up to date version of this documentation. In addition you will find the latest Automic software components.

Automic documentation is written as a series of topics. Each topic begins with a heading followed by a summary paragraph. The summary paragraph states the key concepts presented in the topic. To get a quick overview of a chapter, read the summary paragraph for each topic and look at the images and their captions.

Documentation at docs.automic.com contains the most up to date version of this documentation. It is provided in both .pdf and HTML5 formats.

The HTML5 documentation is displayed as individual HTML pages. You can:

- Find documents in the table of contents.
- Find documents with full-text search.
- Print individual topics.

Contents

The **Contents** tab contains the contents of the documentation in a tree-view of the directory structure. It includes all the guides and their topics.

To expand or collapse a book or topic with subtopics, click it. To view a topic, click it too.

If you click a hyperlink in a topic that takes you to a new topic, the table of contents is refreshed to show your location.

Searching the HTML5

You can search through the entire documentation using the search field in the top right-hand corner of the web page. Enter one or more search terms in the field and start the search either by clicking the magnifying glass icon or using the Enter key. The search results are displayed in a separate screen.

- Uppercase and lowercase letters are not distinguished (not case-sensitive).
- You can use any combination of letters (a-z) and numbers (0-9).
- You can search for several words. The following combinations can be used: AND relations (separators: AND, +, &), OR relations (blank or | can be used as a separator) and negations (^ character in front of the term, NOT will not work).
- To combine search operators and expressions, use ( ). Example: term1 AND (term2 OR term3).
- A search for run will also yield the following results: running, runner and runtime.
- Search results will be highlighted.
- Wildcard characters are not supported.
- Punctuation marks that are used in the word (such as a dot, colon, semi colon, comma or hyphen) have the effect that the searched term is split into two words.

Printing Topics

To print individual topics from a HTML5 documentation, select the topic and click **Print** in the bar at the top of the screen.

To print entire guides, print from the .pdf files.

Using the Compatibility Checker

The Automic Compatibility Checker found at docs.automic.com is the only official and up to date source for compatibility of Automic software components, versions, and sub-components.

2 Working with the RA Web Service SOAP Agent

The basic steps for implementing the Rapid Automation Web Service Agent integration include:

1. Loading the RA Web Service SOAP Agent solution **.jar** file into Automic.
2. Configuring the RA Web Service SOAP Agent in Client 0.
3. Creating RA Web Service SOAP Agent Connection objects.
4. Creating RA Web Service SOAP Agent Jobs.

These steps are described in the following topics in this chapter.

2.1 Installing/Upgrading the Web Service SOAP Agent in an Existing Automation Engine System

This topic describes how to install the Web Service SOAP Agent in an existing Automation Engine system.

Warning: The Web Service Rapid Automation Agent is not available on IBM AIX operating systems.

No Upgrade from v2 or v3 RA Web Service Agent

The RA Web Service SOAP Agent is based on different libs/solution than v3.x. V4.x uses CXF for SOAP Jobs, while v3 used AXIS2. There is no automatic upgrade from Web Service v2 or v3 Agent to Web Service v4 Agent. All connection and Job objects have to be defined/upgraded manually in v4. Web Service Agent v4 solution doesn't replace the Web Service Agent v2 or v3 solution, it needs to be installed on as a separately installed RA Agent. Consequently, a combination of Web Service v2, v3, and v4 Agents can be run in parallel within one Automation Engine or CA Automic Applications Manager system. The RA Web Service SOAP Agent Agent is recommended for new Web Service customers or existing customers who require the CXF libs/solution.

Steps to Install or Upgrade the Web Service SOAP Agent

Warning: Make sure that this and other Java Agents can only connect to **CP** port numbers that are lower than 65536. If they use a higher port number, the Agent cannot start and aborts with an error message. This limitation is caused by Java and affects the Agents for JMX, Databases, SAP, and RA.

To set-up the RA Web Service SOAP Agent, you need to:

- Meet the Java Requirements as described in the Java Requirements section below.
- Load a license in the database for the Agent.
- Put the Agent core file(s) on the host machine.
- If your Automation Engine version is less than 12.1, delete the content of the **bin/lib** directory from the RA core installation that hosts/runs the Agent.
- Do one of the following:
  - **For Windows:** Execute the **setup.exe** file.
  - **For UNIX:** Unpack the **ucxjcitx.tar.gz** file.
- Edit the **ucxjcitx.ini** file.
- Load the Web Service SOAP **.jar** file into the database.
- If you are using the Local Client rather than the Automic Web Interface, you must delete the contents of the **lib** folder of the RA core.
- Create an Agent object. The RA Agent only starts if an Agent object of the same name exists in system client 0000.
- Enable Rapid Automation trace to provide more troubleshooting information.
- Start the Agent.
- Disable Rapid Automation trace.

Java Requirements

On the host, check the current version of your system's Java Virtual Machine (JVM) using the following command:

```
java -version
```

The Web Service SOAP Agent requires Java JDK 1.7 or 1.8 on the Agent machine.

On the Agent machine, the explicit path to JDK Java 1.7 or 1.8 is needed on the command to start the Agent as shown below:

```
/etc/alternatives/jdk1.7.0_45/bin/java  -Xmx2048m  -jar ucxjcitx.jar
disable_cache
```

For platform-specific Java requirements on your Automation Engine machine, see your Automation Engine release notes.

You can get the necessary files to install Java from Oracle.

License File Requirement

License files for RA Agents need to have a **EX.RA.<AGENT TYPE>** line in them and be loaded into the database. For more information on loading keyfiles, see your Automation Engine documentation.

Putting Agent Core Files on the Host Machine

Install each RA Agent in its own sub-directory on its host machine.

Warning: The Agent core must be the same version as the Automation Engine. You can update the Agent core by getting updated files for an Automation Engine release by downloading the image for that release from the Automic Download Center and putting the Agent files in place following these instructions.

For UNIX

1. On the host machine, create a directory for the Agent. We highly recommend installing each RA Agent type on the same host in its own sub-directory. For example:
   ```
   automic/agents/ra_ftp
   automic/agents/ra_bo
   automic/agents/ra_ws
   ```
2. Copy the **ucxjcitx.tar.gz** file in the **\Automation.Platform\Agents\rapidautomation\Core\unix** sub-directory from the Automic Download Center to the sub-directory you created for this Agent.
3. Unpack the **ucxjcitx.tar.gz** file using the following commands:
   ```
   gunzip ucxjcitx.tar.gz
   tar -xvf ucxjcitx.tar
   ```
4. Delete the content of the **lib** folder.

For Windows

1. On the host machine, create a directory for the Agent. We highly recommend installing each RA Agent type on the same host in its own sub-directory. For example:
   ```
   C:\automic\agents\ra_ftp
   C:\automic\agents\ra_bo
   C:\automic\agents\ra_ws
   ```

2. Copy the files in the **\Automation.Platform\Agents\rapidautomation\Core\windows\x86** sub-directory from the Automic Download Center to the sub-directory you created for this Agent.
3. Execute the **setup.exe** file.
4. Delete the content of the **lib** folder.

Supplied Files

The Rapid Automation Agent core includes the following notable files:

- **ucxjcitx.jar**

  Agent core for Rapid Automation

- **ucxjcitx.ini**

  Rapid Automation configuration file

- ***.jar**

  Libraries

- **uc.msl**

  Message library

- **setup.exe**

  The Windows installation executable for the RA Agent core

Editing the ucxjcitx.ini File

Edit the required parameters in the **ucxjcitx.ini** file for the RA Agent described below.

- **name**

  Name of the Agent object. The Agent name is limited to 32 of the following characters: "A-Z", "0-9", "_", ".", "$", "@", "-" and "#".

  Hyphens ("-") are only allowed in Agent names. They must not be used in the names of any other objects.

  Although Agent names are limited to 32 characters, you should keep them under 25 characters. The last seven characters are used for adding the suffix '.NEW.nn' when a new Agent is created from its template.

- **system**

  Automation Engine system name. This entry must be identical to the entry in the **.ini** file of the Automation Engine server.

- **cache_directory**

  Directory to which the Agent should store the RA solutions. This will be set to **cache** by default and does not need to be altered unless you want to change it.

- **lib_directory**

  Directory that contains external libraries that are not part of the solution (such as ojdbc6.jar).

  Default: lib

- **ra**

  Used for additional trace. Before starting a newly installed or upgraded Agent, it is a good idea to turn Rapid Automation trace on by adding **ra=99** as shown below. This will give more troubleshooting information if something goes wrong during the install. After a successful Agent start, you can set **ra=0**, and restart the Agent to turn Rapid Automation trace off.

- **cp**

  Address of the communication process in the Automation Engine system to which the Agent should connect itself. The format is:

  ```
  <DNS name or TCP/IP address>:<port number>
  ```

For information on the additional parameters in the **ucxjcitx.ini** file, see your Automation Engine documentation.

A sample **ucxjcitx.ini** file is shown below. The required parameters are shown in bold:

```
[GLOBAL]
name=RA01
system=AE
logcount=10
logging=../temp/RA_LOG_##.TXT
;LogMaxSize: 0...default, qualifiers k...Kilo, M...Mega, G...Giga
LogMaxSize = 0
language=E
helplib=uc.msl

[RA]
cache_directory=cache
ext_directory=external
shared_directory=shared
lib_directory=lib

[TCP/IP]
connect=20
cp=localhost:2217

[AUTHORIZATION]
KeyStore=
InitialPackage=

[VARIABLES]
uc_host_jcl_var=RA
uc_ex_path_bin=.
uc_ex_path_temp=..\temp\
uc_ex_path_jobreport=..\temp\

[TRACE]
file=..\temp\RA_TRACE_##.TXT
;TraceMaxSize: 0...default, qualifiers k...Kilo, M...Mega, G...Giga
TraceMaxSize=0
tcp/ip=0
ra=99
trccount=10

[CP_LIST]
2218=PC01
```

Loading the Web Service SOAP .jar File into the Database

On the host machine, start the utility **AE.DB Load** and select the RA Web Service SOAP Agent's **.jar** file. The utility will then load it to the Automation Engine database. The **.jar** file can be loaded via the graphical interface or the Java batch mode (**ucybdbld.jar**) of the utility Automation Engine DB Load. Loading with the Automation Engine DB Load in batch mode (**ucybdbld.exe**) under Windows is not possible.

The RA Agent can only connect to one RA solution. If you intend to use several RA solutions, keep in mind that each solution requires its own RA Agent.

You cannot load the same **.jar** file of an RA solution to several systems at a time. Any attempt to do so can cause the utility Automation Engine DB Load to abort.

Using the krb.ini (.conf) File for Kerberos Authentication

Kerberos settings other than **User** and **Password** are set in the **krb.ini** (**.conf**) file. The Agent itself does not read this file directly, it uses the Kerberos classes of the JRE.

The algorithm to locate the **krb5.conf** file is the following:

- If the system property **java.security.krb5.conf** is set, its value is assumed to specify the path and file name.
- If that system property value is not set, the configuration file is looked for in the directory:
  - **<java-home>\lib\security** (Windows)
  - **<java-home>/lib/security** (Solaris and Linux)

  Here **<java-home>** refers to the directory where the JRE is installed. For example, if you have J2SE 5.0 installed on Solaris in a directory named **/j2sdk1.5**, the directory in which the configuration file is looked for is:

  ```
  /j2sdk1.5/jre/lib/security
  ```
- If the file is still not found, then an attempt is made to locate it as follows:
  - **/etc/krb5/krb5.conf** (Solaris)
  - **c:\winnt\krb5.ini** (Windows)
  - **/etc/krb5.conf** (Linux)
- If the file is still not found, and the configuration information being searched for is not the default realm and KDC, then implementation-specific defaults are used. If, on the other hand, the configuration information being searched for is the default realm and KDC because they weren't specified in system properties, and the **krb5.conf** file is not found either, an exception is thrown.

Starting the Web Service SOAP Agent

The Web Service SOAP Agent only starts if an Agent object of the same name exists in system client 0000. A template for the Agent objects is stored in the TEMPLATE folder.

You can use the following command to start the Agent via the command line (UNIX and Windows):

```
/etc/alternatives/jdk1.7.0_45/bin/java  -Xmx2048m  -jar ucxjcitx.jar
disable_cache
```

You can also start the Web Service SOAP Agent using the Service Manager program. For more information, see your Automation Engine documentation.

Warning! Web Service Agents should be started via the **Service Manager** program.

If you load the Web Service SOAP solution, then start the Agent shortly afterward, you may get a cached Agent rather than the one you just loaded. You can avoid this by adding **disable_cache** to the end of the start command. That way the loaded version is always started.

2.2 Specifying Agent Settings

Using the Web Service page, you can determine whether to allow Groovy queries of response data. You can set the password for the certification authority. And you can use the SSL Certificates settings to define and manage a JKS Keystore.

Specifying Settings

The **Settings** section includes the following options.

- **Allow Groovy parsings of response data**

  Allows anyone with access to edit RA Web Service SOAP Agent Jobs to select **Groovy** as a query type and include Groovy code as the input when defining queries. By default this option is unchecked for security purposes. When checked, users can define **Groovy** query types when defining queries from the **Response** page for Jobs. For more information on SOAP responses, see Defining Responses for RA Web Service SOAP Agent Jobs.

- **Use Native mode**

  Allows you to omit a check for the JDK and work without CXF.

  This option is checked by default for new Agents or when you first upgrade to v4.1.0 or above.

Using Encrypted Values in Request XML

When the **Use Native mode** box is checked you can use encrypted values in your SOAP request. There are different ways how this can be done:

1. Use a PromptSet with a text field in your RA Web Service SOAP Agent Job and ensure that the box **Show as Password** is checked for the text field. Use the variable name defined for the text field in your request XML for the encrypted value.
2. Use the same PromptSet attached to a Script object which you can utilize then as a kind of generator for encrypted values. Simply start your script, enter the value you would like to encrypt in the PromptSet and open the activation report afterward. You will find a line such as:
   ```
   2017-08-25 09:39:59 - U00020206 Variable '&PW#' was stored with
   value
   '--106DA50A126F227F15'.
   ```
   Copy the value starting with '--10' to the clipboard and paste it directly into your request XML. Alternatively assign the value to an object variable and use the variable in your request.

Updating the Certification Authority Password

If you updated your certification authority password outside of the RA Web Service SOAP Agent, you need to update it in the **Password** field in the **CA Certificates** section. You must then restart the Agent.

Specifying SSL Certificate Settings

Defining a Keystore

The Keystore holds trusted certificates for client/server authentication. To define a keystore:

1. In the **SSL Certificates** section, enter a keystore and password in the **Keystore** and **Password** fields.

   You can browse to the keystore using the browse icon.

2. Save and close the Agent definition.
3. Stop and restart processes for the RA Web Service SOAP Agent solution Agent.
4. Open your RA Web Service SOAP Agent solution Agent object again and reselect **Web Service** tab.

   The certificates for the keystore will be listed in the **SSL Certificates** table.

5. From the **SSL Certificates** table, you can:
   - Import a certificate
   - Export a certificate
   - Delete a certificate

Importing a Certificate into the Web Service Keystore

To import a **.der** certificate into the Web Service Agent keystore:

1. Click **Import** to open the **Import** dialog.

   On the Import dialog, enter a certificate file and alias in the **Certificate File** and **Certificate Alias** fields.

   You can use the browse icon to browse to the certificate file.

2. Click **Import**.

Warning: You cannot import **.pem** certificates. Additionally, if you have a **.der** certificate that was converted from a **.pem** format, you may not be able to browse to it.

Exporting a Certificate from the Web Service Keystore

To export a certificate from the RA Web Service SOAP Agent keystore, select the certificate in the table and click **Export**. The **Save file on agent file system** dialog will appear. Browse to a or enter an absolute file name you want to export the certificate to, click the **Save** button to start the export.

Deleting a Certificate from the Web Service Keystore

To delete a certificate from the RA Web Service SOAP Agent keystore, select the certificate and click **Delete**.

2.3 Creating RA Web Service SOAP Agent Connection Objects

To link the RA Web Service SOAP Agent to an RA Web Service SOAP Agent Job's web service, you must create a Connection object.

RA Web Service SOAP Agent Connection objects store all the connection information to a service end-point. They define the security protocol and corresponding credential information. RA Web Service SOAP Agent Connection objects also contain the URL of the WSDL. You assign RA Web Service SOAP Agent Connection objects to your RA Web Service SOAP Agent Jobs. RA Web Service SOAP Agent Connection objects can also contain global headers that can be used in your Job definitions.

To create RA Web Service SOAP Agent Connection object:

1. Add a Connection object of type **WEBSERVICESOAP** > **SOAP**.
2. Select the **Web Service** object tab.
3. Specify the WSDL file for the Connection object by either:
   - Typing it into the **WSDL** field.
   - Using the browse icon to browse URL for a WSDL file for this Connection object.

     You can browse for the WSDL file on the Agent machine using the **WSDL** dialog.

   Security authentication or proxy settings may be required to retrieve the WSDL. You specify security settings selecting a security authentication from the authentication field in the authentication panel and specifying its details. For more information on security authentication settings, see Specifying Authentication Settings for RA Web Service SOAP Agent Connection Objects. In some rare cases, you may need to specify authentication settings to retrieve the WSDL, and then later specify different authentication settings for the end URL. For more information on proxy settings, see Using a Proxy Server for RA Web Service SOAP Agent Connection Objects.

   If your WSDL file includes umlauts characters on UNIX/Linux java distribution, a special parameter for **javac** is required. In this case, it's necessary to export an environment variable as shown below so that this settings will be automatically active:

   ```
   export JAVA_TOOL_OPTIONS=-Dfile.encoding=UTF8
   ```
4. Click the **Retrieve** button.

   The RA Web Service SOAP Agent lists all the services defined in the **.wsdl** file in the **Service** field.

5. Select a service from the **Service** field.
6. Select a port from the **Ports** field.

   The RA Web Service SOAP Agent lists all the available methods in the **Operations Preview** box.

7. To override the service endpoint defined in the **.wsdl** file, check the **Overwrite, URL Endpoint** checkbox and enter a new endpoint in the **URL Endpoint** field.
8. Optionally specify security settings using the **Authentication** section of the page. For more information, see Specifying Authentication Settings for RA Web Service SOAP Agent Connection Objects.
9. Optionally add HTTP headers to the Connection object to pre-populate them in Jobs using the **HTTP Request Headers** section of the page. For more information,,see Defining HTTP Headers for RA Web Service SOAP Agent Connection Objects.

10. Optionally add SOAP headers to the Connection object to pre-populate them in Jobs using the **SOAP Request Headers** section of the page. For more information, see Defining SOAP Headers for RA Web Service SOAP Agent Connection Objects.
11. Optionally specify proxy settings using the **Proxy** section of the page. For more information, see Using a Proxy Server for RA Web Service SOAP Agent Connection Objects.
12. Optionally specify advanced settings on using the **Advanced** section of the page. For more information, see Specifying Advanced Settings for RA Web Service SOAP Agent Connection Objects.
13. Click **Save**.

The **Login** field on the **Attributes** page is not used for Automation Engine Jobs of the Web Service SOAP Agent.

2.3.1 Specifying Authentication Settings for RA Web Service SOAP Agent Connection Objects

Security settings for RA Web Service SOAP Agent Connection objects can be set in the **Authentication** section

Some **.wsdl** files require authentication specifications to the end URL. To enter them:

1. Select an option from the **Authentication** field.

   Unless you select "None" for the authentication, additional authentication-specific fields will be added below the **Authentication** drop-down list, based on the requirements and options for that authentication.

2. Respond to the authentication-specific fields. Although not all of these fields are required by the RA Web Service SOAP Agent, they may need values for the authentication mechanism to work.

   The settings in the Authentication section differ depending whether the Use Native mode box is unchecked in the Agent definition.

   - When **Use Native mode** is checked, the following authentications are available.

For	Uses	Field	Description
Basic and Digest	HTTP Client Credentials (UsernamePasswordCredentials)	**Username**	The user name.
		Password	The password.
NTML	HTTP Client Credentials (NTCredentials)	**Username**	The user name.
		Password	The password.
Kerberos*	The existing kerberos impl. code. we set the Host if its empty per default the host name from URL Endpoint	**Username**	The user name.
		Password	The password.
		Host	The host to connect to. Only required if the endpoint differs from the authentication host.
		URL Endpoint	A read-only URL endpoint

     * Additional configuration settings for Kerberos are set in the **krb.ini** (**.conf**) file, see Additional Kerberos Configuration.

     In native mode for authentications, we have rewritten the code and we use **CloseableHttpClient + BasicCredentialsProvider**.

   - When **Use Native mode** is not checked, the following authentications are available.

For	Field	Description
Basic, and NTLM	**Username**	The user name.
	Password	The password.
	Host	The host to connect to. Only required if the endpoint differs from the authentication host.
	Port	The port to use. Only required if the endpoint differs from the authentication host.
	Realm	The realm. Only required if the endpoint differs from the authentication host. Also, this field is optional when a host is specified.
	Preemptive	Activates preemptive authentication. Preemptive authentication sends the authentication information without waiting for the server to give an unauthorized response. This reduces some overhead, and may be required in cases where the server does not reply with an unauthorized response. You use preemptive authorization when you trust the endpoint enough to send authentication credentials somewhere without being asked for them.
Digest	**Username**	The user name.
	Password	The password.
	Host	The host to connect to. Only required if the endpoint differs from the authentication host.
	Port	The port to use. Only required if the endpoint differs from the authentication host.
	Realm	The realm. Only required if the endpoint differs from the authentication host. Also, this field is optional when a host is specified.
Kerberos*	**Username**	The user name.
	Password	The password.

* Additional configuration settings for Kerberos are set in the **krb.ini** (**.conf**) file, see Additional Kerberos Configuration.

Additional Kerberos Configuration

Additional configuration settings for Kerberos are set in the **krb.ini** (**.conf**) file. The Agent itself does not read this file directly, it uses the Kerberos classes of the JRE.

The algorithm to locate the **krb5.conf** file is the following:

- If the system property **java.security.krb5.conf** is set, its value is assumed to specify the path and file name.
- If that system property value is not set, the configuration file is looked for in the directory:
    - **<java-home>\lib\security** (Windows)
    - **<java-home>/lib/security** (Solaris and Linux)

Here **<java-home>** refers to the directory where the JRE is installed. For example, if you have J2SE 5.0 installed on Solaris in a directory named **/j2sdk1.5**, the directory in which the configuration file is looked for is:

```
/j2sdk1.5/jre/lib/security
```

- If the file is still not found, then an attempt is made to locate it as follows:
  - **/etc/krb5/krb5.conf** (Solaris)
  - **c:\winnt\krb5.ini** (Windows)
  - **/etc/krb5.conf** (Linux)
- If the file is still not found, and the configuration information being searched for is not the default realm and KDC, then implementation-specific defaults are used. If, on the other hand, the configuration information being searched for is the default realm and KDC because they weren't specified in system properties, and the **krb5.conf** file is not found either, an exception is thrown.

2.3.2 Defining HTTP Headers for RA Web Service SOAP Agent Connection Objects

When you define a new RA Web Service SOAP Agent Job using a Connection object with headers defined, the Job will be pre-populated with the headers. It is also possible to add additional headers to Job definitions, or choose to only use headers in Job definitions that are defined there.

You define HTTP headers for an RA Web Service SOAP Agent Connection object using the **HTTP Headers** section.

2.3.3 Defining SOAP Headers for RA Web Service SOAP Agent Connection Objects

When you define a new RA Web Service SOAP Agent Job using a Connection object with SOAP headers defined and you are not using native mode, the Job will be pre-populated with the headers.

How SOAP Headers are defined differs depending whether the Use Native mode box is unchecked in the Agent definition.

- When **Use Native mode** is checked, The SOAP header table inputs are ignored from the Connection object are ignored.
- When **Use Native mode** is not checked, **SOAP Headers** are available in RA Web Service SOAP Agent Jobs' definitions. When you define a new RA Web Service SOAP Agent Job using a Connection object with headers defined, the Job will be pre-populated with the headers. It is also possible to add additional headers to Job definitions, or choose to only use headers in Job definitions that are defined there.

You define SOAP headers for an RA Web Service SOAP Agent Connection object using the **SOAP Headers** section

2.3.4 Using a Proxy Server for RA Web Service SOAP Agent Connection Objects

If you are using a proxy server, you can specify its settings using the **Proxy** section.

Proxy fields are described below.

Field	Description
Server Type	Allows you to select HTTP or SOCKS as server type or to leave the default of No Proxy to not use a proxy server.
Host	The host where the proxy server resides.
Port	The port of the proxy server.
User Name	The user to pass to the proxy server.
Password	The password to pass to the proxy server.
Realm	The realm. This field is only available when the Use Native mode box is unchecked in the Agent definition.
Authentication	The authentication, options are: • None • Basic • Digest • NTLM This field is only available when the Use Native mode box is checked in the Agent definition and HTTP is selected in the **Server Type** field.

2.3.5 Specifying Advanced Settings for RA Web Service SOAP Agent Connection Objects

You define advanced setting an RA Web Service SOAP Agent Connection object using the **Advanced** section.

Advanced fields are described below.

Field	Description
Connection Timeout	The number of seconds before timing out while attempting to connect to the URL endpoint. When set to 0, the connection never times out. The default is 30 seconds.
Read Timeout	The number of seconds before timing out when waiting for a method call. When set to 0, the read never times out. The default is 60 seconds.

2.4 Creating RA Web Service SOAP Agent Jobs

To run a SOAP web service, you need to create an RA Web Service SOAP Agent Job.

To create an RA Web Service SOAP Agent Job:

1. Add a Job object of type **WEBSERVICESOAP** > **SOAP** and select your RA Web Service SOAP Agent in the **Host** field on the **Attributes** page.
2. Select the **Web Service** tab.
3. In the **General** and **URL Endpoint and reporting** sections, respond to the fields described below.

Field	Description
Connection	A RA Web Service SOAP AgentConnection object. The Connection object you select can connect to any RA Web Service SOAP Agent solution Agent object that's available on the client. For more information on RA Web Service SOAP Agent Connection objects, see topic Creating RA Web Service SOAP Agent Connection Objects.
Service	A non-editable field that shows the service for the selected Connection object.
Port	A non-editable field that shows the port for the selected Connection object.
Operation	The available methods for the web service. When you select an operation, XML for the request is written to the Job definition. You view and edit the XML by selecting the **Request > XML** menu item on the left side of the screen. If you change the operation, the existing XML including any edits you have made will be erased. Note that you will get a confirmation pop-up message allowing you to confirm the change when you select a new operation.
Include required fields only in generated XML request	When selected, only required fields will be written to the XML in the Job definition when you select an operation in the **Operation** field above.

Remove template values from generated XML request	When selected, the default template values will be removed from the XML when you select an operation in the **Operation** field above. In the example below, **REQUEST_VALUE_symbol** is automatically written as a template value for Strings.
	
	`<symbol>REQUEST_VALUE_symbol</symbol>`
	
	When this box is checked, the template value **REQUEST_VALUE_symbol** is removed as shown below.
	
	`<symbol></symbol>`

Overwrite URL Endpoint	Allows the Job to override the URL Endpoint from the connection. When checked the **URL Endpoint** field below becomes editable.

Field	Description
URL Endpoint	An optional field where you can override the URL endpoint defined in the Connection object. When this field is non-editable (the **Overwrite URL Endpoint** checkbox is not selected), the **Connection URL Endpoint** is shown. When the Overwrite URL endpoint checkbox is selected, the field becomes editable and changes to the selected Connection object will have no effect on this Jobs URL Endpoint.
Write request and response to log	Includes the XML for the request and response in the Job log.
Create XML SOAP request report	Creates a Job report the XML for the SOAP request. These reports are written to the **soap_reports** directory.
Create XML SOAP response report	Creates a Job report with the XML for the SOAP response. These reports are written to the **soap_reports** directory.

4. Specify request and response settings using the menu items on the left side of the screen as documented in the following topics.

Deleting Registered Files

Registered output files for RA Web Service SOAP Agent Jobs are saved to the **task_reports** directory under the **bin** directory. They will remain there until you move or delete them.

2.4.1 Defining Requests for RA Web Service SOAP Agent Jobs

Use the Request HTTP/SOAP Headers section to define HTTP and/or SOAP headers for RA Web Service SOAP Agent Jobs. Headers can be defined both as defaults for the Connection object, or in individual Jobs. You can edit the values of default Connection object headers in the Job definition, or delete them. Select the Request XML and Attachments to edit the SOAP XML for a request objects and define attachments.

Defining HTTP and SOAP Request Headers

The Headers available on the **Request** and **Response** pages differ depending whether the Use Native mode box is unchecked in the Agent definition.

- When **Use Native mode** is checked, only **HTTP Request Headers** are available.
- When **Use Native mode** is not checked, both **HTTP Request Headers** and **SOAP Headers** are available, because the header is part of the envelope.

Headers can be defined both as defaults for the Connection object, or in individual Jobs. You can edit the values of default Connection object headers in the Job definition, or delete them.

Viewing, Editing or Deleting Default Headers From a Job's Connection Object

Default headers that are defined in a Job's Connection object are included in the **HTTP Headers** and/or **SOAP Headers** tables in gray italics. They are read-only, unless the **Use Job HTTP Headers Only** or **Use Job SOAP Headers Only** checkboxes are checked. If one of these checkboxes is checked, all headers defined in the Connection object will remain in the table and their backgrounds switch to white like the headers defined for the Job and they will become editable or deletable.

Working in the HTTP and SOAP Headers Tables

To add a row to the table, click **Add Row** in the appropriate section and click the cells in the table to edit their values

Defining Request XML

To edit the SOAP XML for a request object:

1. Select the **Request** tab on the screen.
2. Edit the Job request code.

   If you change the operation on the main **Web Service** page, the existing XML including any edits you have made will be erased. Note that you will get a confirmation pop-up message allowing you to confirm the change when you select a new operation.

   When the **Use Native mode** box is unchecked in the Agent definition, you can include WSDL information in the request.

Setting the Language in SOAP Message Requests

For SOAP clients, the language must be specified as BYTE depending on encoding. For example, 68 for German or 69 for English as shown below.

```
<language>69</language>
```

Defining Request Attachments

RA Web Service SOAP Agent Job definitions include request attachments.

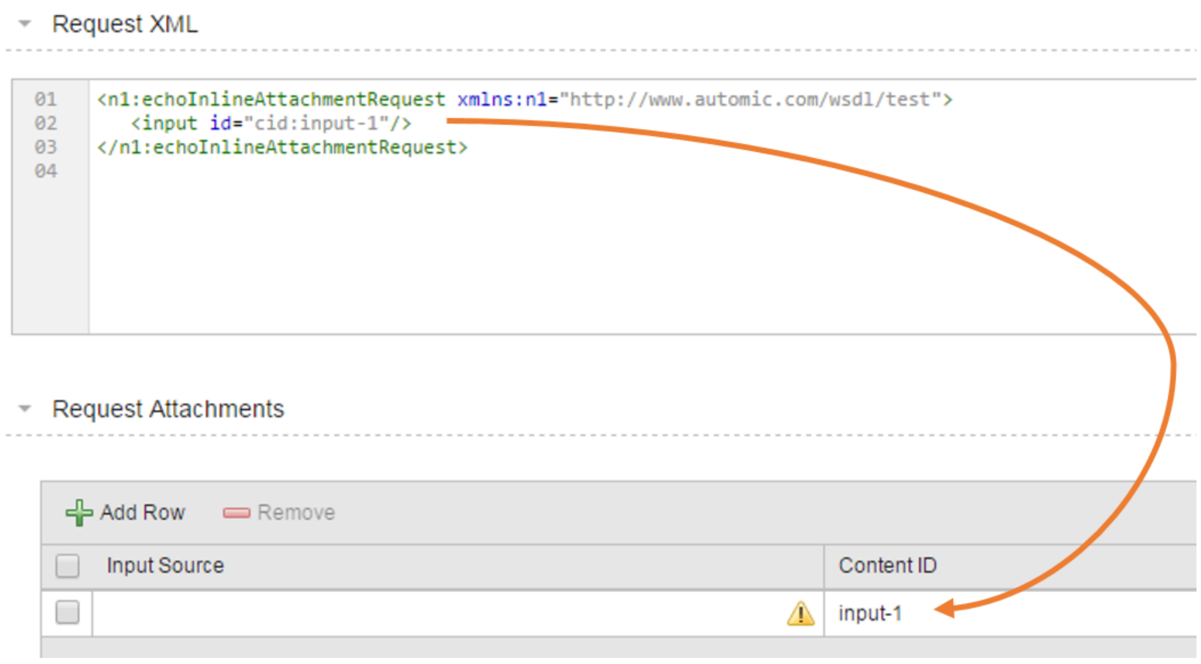
Working in the Attachments Table

If the Agent detects attachment references (data types base64Binary, hexBinary or swaRef) in the request XML, this table will be pre-populated when the operation is selected.

You can add additional rows if the request includes a list of attachments or if you want to add attachments which have no reference in the request XML (SWA attachments).

Instead of specifying a file path and name in the table, you can alternately add the attachment directly to the request XML, but you must encode it correctly (e.g. base64 encoded for an element of type base64Binary).

Identification of attachment is not based on id attribute placed in the binary or hexBinary element in request XML. The id attribute is added automatically by retrieving the WSDL and contains the reference to the attachment table. Each attachment identifier must contains the id attribute and the value have to start with **cid:** string, or else it is not a valid identifier and will not be processed by the Job.



```
<n1:echoInlineAttachmentRequest xmlns:n1="http://www.automic.com/wsdl/test">
   <input id="cid:input-1"/>
</n1:echoInlineAttachmentRequest>
```

Input Source	Content ID
	input-1

In CXF mode, the attachments has been linked to XML node name which was related to the Content-ID with same name, like "Data". The problem was that the attachments has been processed sequentially, regarding of the name - vs. Content-ID.

To add a row to the table, click **Add Row** in the **Attachments** section and click the cells in the table to edit their values

Enabling MTOM or XOP

To enable MTOM or XOP attachments, select the **MTOM** or **XOP** option for the **Transfer type**. When **MTOM** is selected, the attachment is referenced from within the SOAP envelope by an href attribute and is not base64 encoded. The attachment data is outside the SOAP envelope. This is considered to have better performance than SWA. Base64 encoding makes the attachment size 33 percent larger. XOP is similar to MTOM. The only difference is, that XOP wraps the reference to the attachment in own XML node to indicate the XOP processing.

2.4.2 Defining Responses for RA Web Service SOAP Agent Jobs

You can define HTTP and/or SOAP response headers, extract data from a SOAP XML responses by defining parsings, and define response attachments.

Defining HTTP and/or SOAP Response Headers

The Headers available on the **Request** and **Response** pages differ depending whether the Use Native mode box is unchecked in the Agent definition.

- When **Use Native mode** is checked, only **HTTP Request Headers** are available, because the header is part of the envelope.
- When **Use Native mode** is not checked, both **HTTP Request Headers** and **SOAP Headers** are available.

You can define a Response Header to extract a value from the headers returned rather than from the response content. For example, you may have a call that returns a token value in a response header that you want to extract to use in subsequent Jobs.

Working in the HTTP and SOAP Headers Tables

To add a row to the table, click **Add Row** in the appropriate section. Click the cells in the table to edit their values

HTTP and SOAP Headers Column Descriptions

Column	Description
Header Name	The HTTP or SOAP header name.

Column	Description
Output Type	Enter an output type in the **Output Type** field. Available options are: • **Save as Variable:** A variable with the name you specify in the **Output Name** field is saved to the database. For Automation Engine v9/10, if the size is: ◦ Less than 1024 bytes, the value is included in the database. ◦ Greater than 1024 bytes, the value is written to **<Agent>/bin/variableFileDirectory**. An Automation Engine scripting language variable's name is limited to 32 alphanumeric characters, including the special characters "$", "_", "@", "§" and "#". German Umlauts are not allowed. The first character must not be a number. • **Write to Job Report:** Writes the resulting value to the Job's report log based on the name you specified in the **Output Name** field. • **Save as Report:** Saves the resulting value to a report in the **<Agent>/bin/task_reports** directory based on the name you specified in the **Output Name** field. The file will be registered with the Automation Engine and viewable from the UserInterface. When this option is used, the Web Service Agent expects a parameter value of just a stand-alone file name. If you enter a file name with fully qualified path, the Job will get an error like the following: `java.io.FileNotFoundException: task_reports/u01/users/qa4/v9/RA_WS_Alpha/test3.txt (No such file or directory)` • **Write to File:** Writes the resulting value to a file in the Agent's file system. You can enter either a stand-alone file name to write to the **<Agent>/bin/task_reports** directory on the Agent's file system or a file name with fully qualified path to a different location on Agent's file system. The file will not be registered with the Automation Engine and is not viewable from the user interface.
Output Name	The output name.

Defining Response Parsings and Attachments for RA Web Service SOAP Agent Jobs

You can create a response parsing query to extract values from the response. Additionally, you can define response attachments.

Working in the Response Parsing and Attachments Tables

To:	Do this:
Add a row to the table	Click **Add Row**in the appropriate section. Click the cells in the table to edit their values.
Edit a row in the table	Click the cell in the table and edit its value.
Delete one of more rows to the table	Check the checkbox for the row(s) and click **Remove**.
Delete all rows to the table	Check the checkbox column header and click **Remove**.

Response Parsing Column Descriptions

Column:	Description:
Parsing Type	Enter a type in the **Parsing Type** field The options are: • XPath • XQuery • Groovy **Warning:** The Groovy option allows you to harness the power of Java and Groovy to parse the response. Because it makes all the Java/Groovy methods available, it can be a security concern and is therefore turned off by default. Groovy parsing is only done when the Allow Groovy parsings of response data box on the **Settings** section of the **Web Service** page of the Agent is checked. If you need to define Groovy parsings, consult your Automation Engine Administrator.
Output Type	Enter an output type in the **Output Type** field. Available options are: • **Save as Variable:** A variable with the name you specify in the **Output Name** field is saved to the database. For Automation Engine v9/10, if the size is: • Less than 1024 bytes, the value is included in the database. • Greater than 1024 bytes, the value is written to **<Agent>/bin/variableFileDirectory**. An Automation Engine scripting language variable's name is limited to 32 alphanumeric characters, including the special characters "$", "_", "@", "§" and "#". German Umlauts are not allowed. The first character must not be a number. • **Write to Job Report:** Writes the resulting value to the Job's report log based on the name you specified in the **Output Name** field. • **Save as Report:** Saves the resulting value to a report in the **<Agent>/bin/task_reports** directory based on the name you specified in the **Output Name** field. The file will be registered with the Automation Engine and viewable from the UserInterface. When this option is used, the Web Service Agent expects a parameter value of just a stand-alone file name. If you enter a file name with fully qualified path, the Job will get an error like the following: `java.io.FileNotFoundException: task_reports/u01/users/qa4/v9/RA_WS_Alpha/test3.txt (No such file or directory)` • **Write to File:** Writes the resulting value to a file in the Agent's file system. You can enter either a stand-alone file name to write to the **<Agent>/bin/task_reports** directory on the Agent's file system or a file name with fully qualified path to a different location on Agent's file system. The file will not be registered with the Automation Engine and is not viewable from the user interface.
Output Name	The output name.

Column:	Description:
Expression	With XPath and XQuery types, you can enter a value in the **Input** box by doing one of the following: • To create an expression and replace what's in the **Expression** box, right-click a value and pick **Create Expression and Replace**. • To add it at the cursor location in the **Expression** box, select a value and click the **Create Expression** box or right-click a value and pick **Create Expression and Insert**.
Use Array	When this checkbox is checked, the results are saved to an Automation Engine Array variable, where each value can be referenced numerically. For example: &VAR[1], &VAR [2], etc. If the **Save as Variable** checkbox is not selected, the whole result is returned as one variable.

Response Attachments Descriptions

If the Agent detects attachment references (data types base64Binary, hexBinary or swaRef) in the response definition, this table will be pre-populated when the operation is selected.

You can add additional rows if the response includes a list of attachments or if you want to handle response attachments which have no reference in the response XML (SWA attachments).

Instead of specifying a file path and name in the table, you can alternately add the attachment directly to the request XML, but you must encode it correctly (e.g. base64 encoded for an element of type base64Binary).

An attachment is referenced by the "Content ID" that you must know before defining the request.

In CXF mode, the attachments has been linked to XML node name which was related to the Content-ID with same name, like "Data". The problem was that the attachments has been processed sequentially, regarding of the name - vs. Content-ID.

Column:	Description:
Output Type	Enter an output type in the **Output Type** field. Available options are: • **Save as Variable:** A variable with the name you specify in the **Output Name** field is saved to the database. For Automation Engine v9/10, if the size is: • Less than 1024 bytes, the value is included in the database. • Greater than 1024 bytes, the value is written to **<Agent>/bin/variableFileDirectory**. An Automation Engine scripting language variable's name is limited to 32 alphanumeric characters, including the special characters "$", "_", "@", "§" and "#". German Umlauts are not allowed. The first character must not be a number. • **Write to Job Report:** Writes the resulting value to the Job's report log based on the name you specified in the **Output Name** field. • **Save as Report:** Saves the resulting value to a report in the **<Agent>/bin/task_reports** directory based on the name you specified in the **Output Name** field. The file will be registered with the Automation Engine and viewable from the UserInterface. When this option is used, the Web Service Agent expects a parameter value of just a stand-alone file name. If you enter a file name with fully qualified path, the Job will get an error like the following: `java.io.FileNotFoundException: task_reports/u01/users/qa4/v9/RA_WS_Alpha/test3.txt (No such file or directory)` • **Write to File:** Writes the resulting value to a file in the Agent's file system. You can enter either a stand-alone file name to write to the **<Agent>/bin/task_reports** directory on the Agent's file system or a file name with fully qualified path to a different location on Agent's file system. The file will not be registered with the Automation Engine and is not viewable from the user interface.
Output Name	The output name.
Content ID	Either the tag name in the XML or the SWA attachment ID

2.4.3 Creating XML Request Parts Dynamically Via Automation Engine Variables for RA Web Service SOAP Agent Jobs

You create dynamic values for RA Web Service SOAP Agent Jobs by building XML content as a string on an RA Web Service SOAP Agent Job's Pre-Process page, placing &xml# in XML request, and generating the XML request.

The steps for creating dynamic values for RA Web Service SOAP Agent Jobs are described below:

1. Build XML content as a string like the following on an RA Web Service SOAP Agent Job's **Pre-Process** page:

```
:DEFINE &myarr#,string,3
:SET &myarr#[1] = "Roger"
:SET &myarr#[2] = "George"
:SET &myarr#[3] = "Kristen"

:SET &count# = 1
:SET &max# = LENGTH(&myarr#[])
:SET &xmlsnippet# = ""
:WHILE &count# <= &max#
:  SET &xmlsnippet# = STR_CAT(&xmlsnippet#, "<ArrayItem>")
:  SET &xmlsnippet# = STR_CAT(&xmlsnippet#, &myarr#[&count#])
:  SET &xmlsnippet# = STR_CAT(&xmlsnippet#, "</ArrayItem>")
:  SET &count# = &count# + 1
```

2. Place the variable from your **Pre-Process** page(in this case **&xmlsnippet#**) in XML request as shown in bold below:

```
< ?xml version="1.0" encoding="UTF-8" standalone="yes"?>
< stringArray xmlns="http://test.automic.com">
&xmlsnippet#
< /stringArray>
```

3. Run the Job to generate the XML request.

   Sample array values from the **Pre-Process** page the are shown in bold below.

```
< ?xml version="1.0" encoding="UTF-8" standalone="yes"?>
< stringArray xmlns="http://test.automic.com">
<ArrayItem>Roger</ArrayItem>
<ArrayItem>George</ArrayItem>
<ArrayItem>Kristen</ArrayItem>
< /stringArray>
```

2.4.4 Examples: Web Service Response Parsing

Below are examples for different types of parsings for you to try out. The returned result is a String, it may be a single word, a number, or multiple words. In any case, the whole result is used. Therefore, you might want to first define a parsing that returns multiple results, and then fine tune it to only return a single result. For regular expression parsing examples, see a regular expression document.

JSONPath

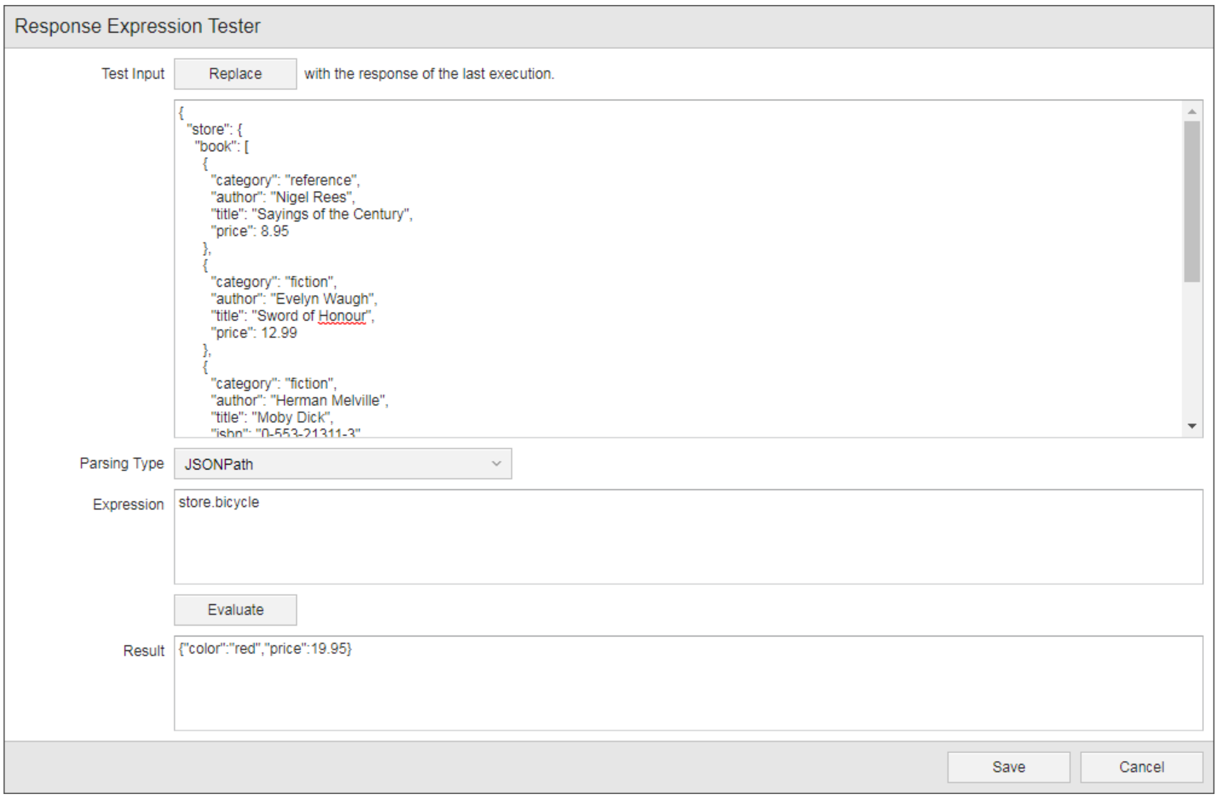
Select **JSONPath** in the **Parsing Type** field and copy the following text and paste it into a parsing's response area.

```
{ "store": {
   "book": [
    { "category": "reference",
      "author": "Nigel Rees",
      "title": "Sayings of the Century",
      "price": 8.95
    },
    { "category": "fiction",
      "author": "Evelyn Waugh",
      "title": "Sword of Honour",
      "price": 12.99
    },
    { "category": "fiction",
      "author": "Herman Melville",
      "title": "Moby Dick",
      "isbn": "0-553-21311-3",
      "price": 8.99
    },
    { "category": "fiction",
      "author": "J. R. R. Tolkien",
      "title": "The Lord of the Rings",
      "isbn": "0-395-19395-8",
      "price": 22.99
    }
  ],
    "bicycle": {
      "color": "red",
      "price": 19.95
    }
  }
}
```

Put each of the following expressions in the **Expressions** box and click **Test**:

- store.book[0]
- store.book[0].title
- store.bicycle
- store.bicycle.color

The parsing results will be displayed in the **Result** box as shown below.


Response Expression Tester
Test Input
Replace
with the response of the last execution.
Parsing Type
Expression
Evaluate
Result
Save
Cancel

```
{
 "store": {
  "book": [
   {
    "category": "reference",
    "author": "Nigel Rees",
    "title": "Sayings of the Century",
    "price": 8.95
   },
   {
    "category": "fiction",
    "author": "Evelyn Waugh",
    "title": "Sword of Honour",
    "price": 12.99
   },
   {
    "category": "fiction",
    "author": "Herman Melville",
    "title": "Moby Dick",
```

Parsing Type: JSONPath

Expression: store.bicycle

Result: {"color":"red","price":19.95}

XPath

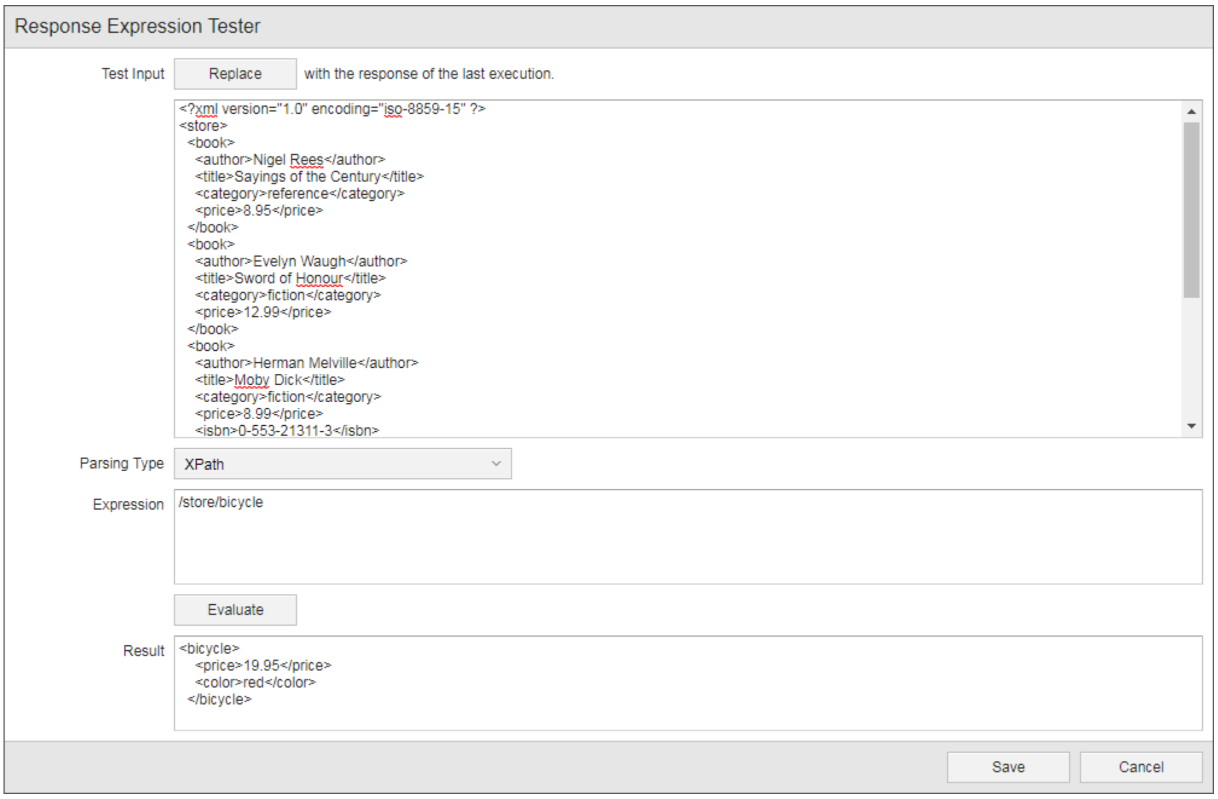
Select **XPath** in the **Parsing Type** field and copy the following text and paste it into a parsing's response area.

```
<?xml version="1.0" encoding="iso-8859-15" ?>
<store>
  <book>
    <author>Nigel Rees</author>
    <title>Sayings of the Century</title>
    <category>reference</category>
    <price>8.95</price>
  </book>
  <book>
    <author>Evelyn Waugh</author>
    <title>Sword of Honour</title>
    <category>fiction</category>
    <price>12.99</price>
  </book>
  <book>
    <author>Herman Melville</author>
    <title>Moby Dick</title>
    <category>fiction</category>
    <price>8.99</price>
    <isbn>0-553-21311-3</isbn>
  </book>
  <book>
    <author>J. R. R. Tolkien</author>
    <title>The Lord of the Rings</title>
    <category>fiction</category>
    <price>22.99</price>
    <isbn>0-395-19395-8</isbn>
  </book>
  <bicycle>
    <price>19.95</price>
    <color>red</color>
  </bicycle>
</store>
```

Put each of the following expressions in the **Expressions** box and click **Test**:

- /store/book
- /store/bicycle
- /store/book[1]/title

The parsing results will be displayed in the **Result box** as shown below.


Response Expression Tester
Test Input
Replace
with the response of the last execution.

```
<?xml version="1.0" encoding="iso-8859-15" ?>
<store>
 <book>
  <author>Nigel Rees</author>
  <title>Sayings of the Century</title>
  <category>reference</category>
  <price>8.95</price>
 </book>
 <book>
  <author>Evelyn Waugh</author>
  <title>Sword of Honour</title>
  <category>fiction</category>
  <price>12.99</price>
 </book>
 <book>
  <author>Herman Melville</author>
  <title>Moby Dick</title>
  <category>fiction</category>
  <price>8.99</price>
  <isbn>0-553-21311-3</isbn>
```

Parsing Type: XPath

Expression: /store/bicycle

Result:

```
<bicycle>
  <price>19.95</price>
  <color>red</color>
 </bicycle>
```

XQuery/JSON XQuery

Query examples using the same **JSON/xml** as the **JSONPath/XPath** examples:

Example 1

```
for $x in $input/store/book
  return $x
```

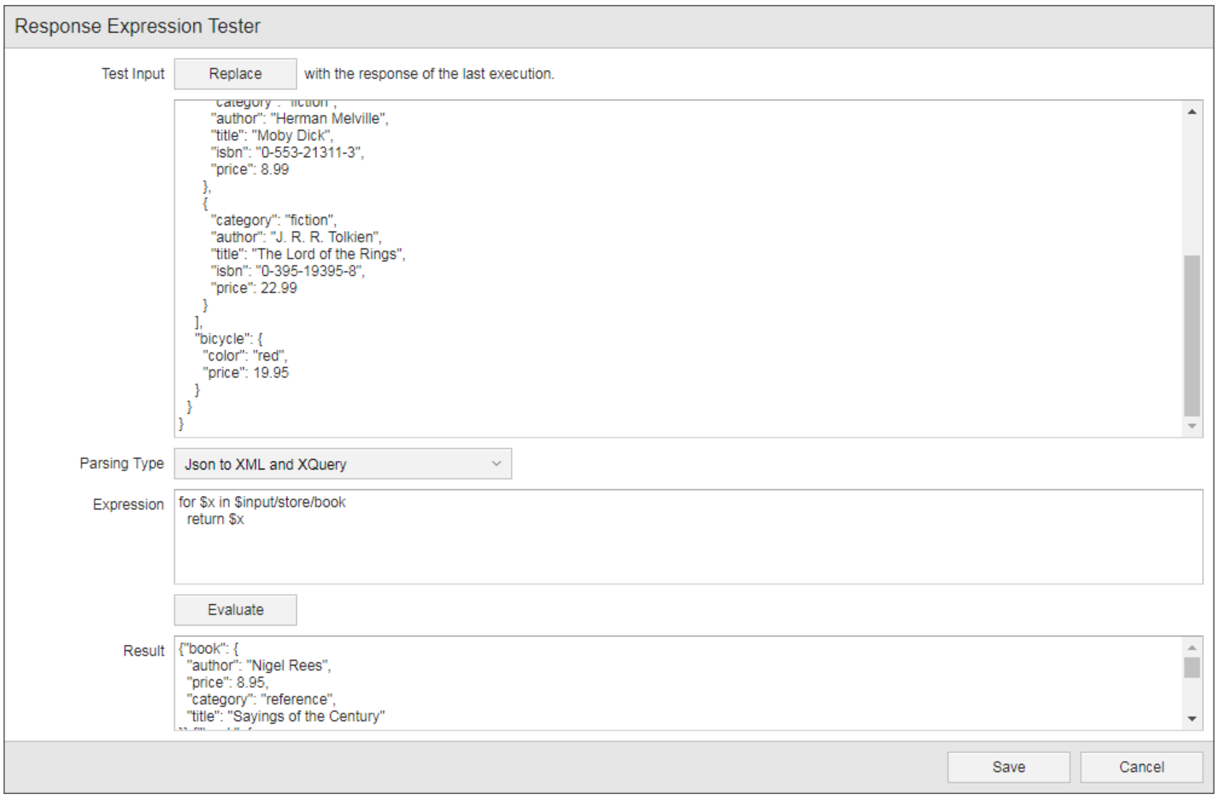
Example 2

```
for $x in $input/store/book[1]
  return $x/title
```

Example 3

```
for $x in $input/store/book[1]
  return $x/title/text()
```

The parsing results will be displayed in the **Result box** as shown below.



```
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
        "price": 8.99
      },
      {
        "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99
      }
    ],
    "bicycle": {
      "color": "red",
      "price": 19.95
    }
  }
}
```

```
for $x in $input/store/book
  return $x
```

```
{"book": {
  "author": "Nigel Rees",
  "price": 8.95,
  "category": "reference",
  "title": "Sayings of the Century"
```

Groovy

Groovy scripts are passed three variables:

- input: The response text
- path: The JsonSlurper.parseText() or XmlSlurper.parseText() result
- logger: used to write a message to the Job report

Path example:

```
def title = path.store.book[0].title
def author = path.store.book[0].author
```

Input example:

```
int n = input.indexOf ('bicycle')
return input.substring (n)
```

logger examples:

Example 1

```
String msg
logger.logMsg (msg)
```

Example 2

```
logger.logMsg ('test message')
```

JSON

A JSON example for the JsonSlurper from the previous examples would be:

```
path.store.book[0].title
```

XQuery/JSON to XML and XQuery

There is no output conversion when the parsing type is set to XQuery/JSON to XML and XQuery.

Example 1: JSON Array with an Array Name

In this case, JSON is converted to XML and parsed with XQuery.

If JSON Response is JSON Array, like the following example, it cannot be converted to XML, because in this case there is no root node specified.

```
{"properties":[{"stringProperty":"prop1","intProperty":"1"},
{"stringProperty":"prop2","intProperty":"2"},
{"stringProperty":"prop3","intProperty":"3"}]}
```

This is the root node in the generated XML.

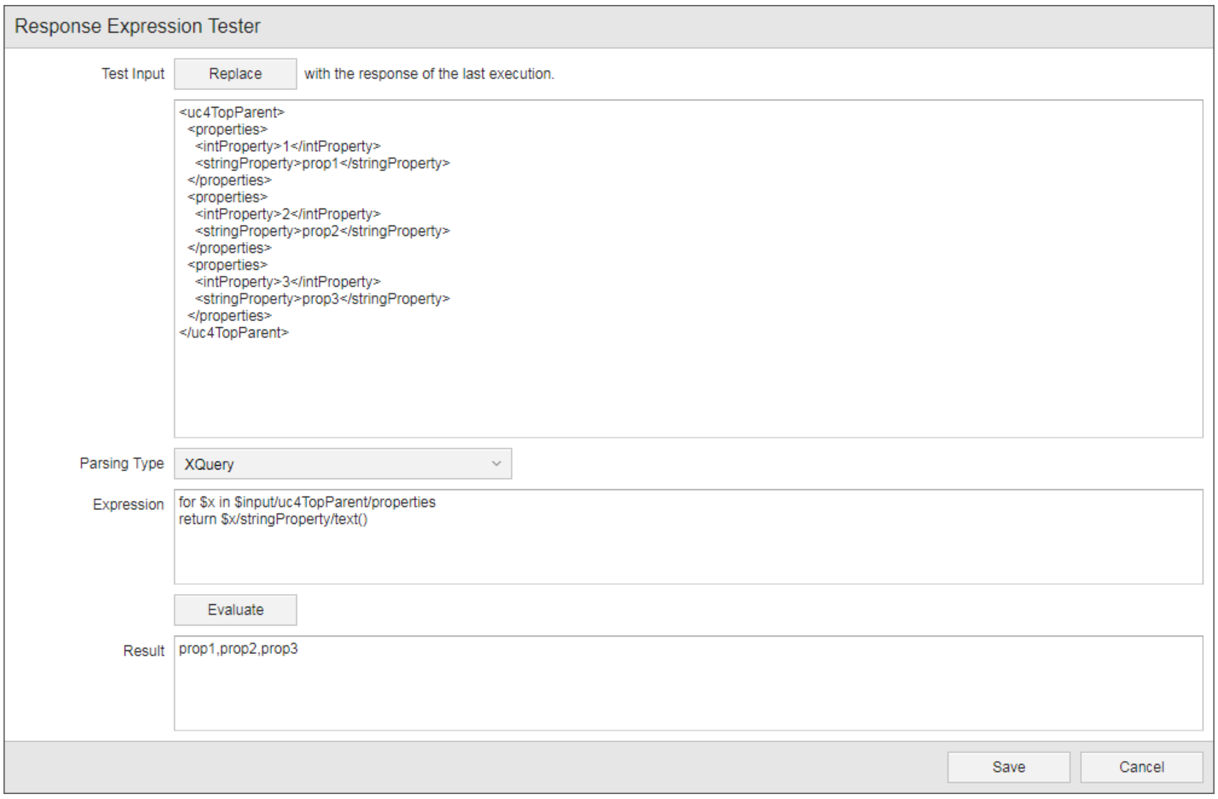
```
<?xml version="1.0" encoding="UTF-8"?>
<uc4TopParent>
   <properties>
      <intProperty>1</intProperty>
      <stringProperty>prop1</stringProperty>
   </properties>
   <properties>
      <intProperty>2</intProperty>
      <stringProperty>prop2</stringProperty>
   </properties>
   <properties>
      <intProperty>3</intProperty>
      <stringProperty>prop3</stringProperty>
   </properties>
</uc4TopParent>
```

In this case, XQuery should include the root node **uc4TopParent** in its expression.

For example XQuery:

```
for $x in $input/uc4TopParent/properties
return $x/stringProperty/text()
```

The parsing results will be displayed in the **Result box** as shown below.



```
<uc4TopParent>
  <properties>
    <intProperty>1</intProperty>
    <stringProperty>prop1</stringProperty>
  </properties>
  <properties>
    <intProperty>2</intProperty>
    <stringProperty>prop2</stringProperty>
  </properties>
  <properties>
    <intProperty>3</intProperty>
    <stringProperty>prop3</stringProperty>
  </properties>
</uc4TopParent>
```

```
for $x in $input/uc4TopParent/properties
return $x/stringProperty/text()
```

```
prop1,prop2,prop3
```

Example 2: JSON Array with No Array Name

In this case, the JSON Response is JSON Array with no array name, like the following:

```
[{"stringProperty":"prop1","intProperty":"1"},
{"stringProperty":"prop2","intProperty":"2"},
{"stringProperty":"prop3","intProperty":"3"}]
```

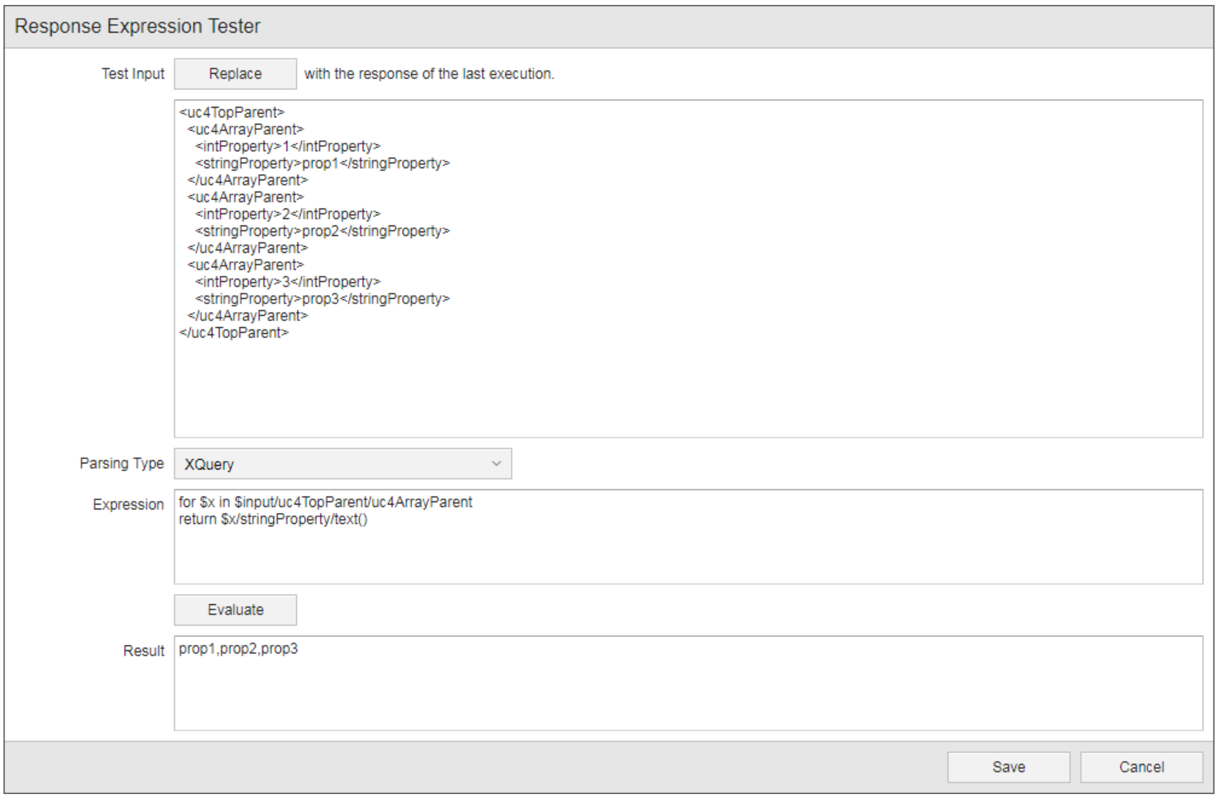
It cannot be converted to XML. In this case array name and root node are generated and XML looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<uc4TopParent>
   <uc4ArrayParent>
      <intProperty>1</intProperty>
      <stringProperty>prop1</stringProperty>
   </uc4ArrayParent>
   <uc4ArrayParent>
      <intProperty>2</intProperty>
      <stringProperty>prop2</stringProperty>
   </uc4ArrayParent>
   <uc4ArrayParent>
      <intProperty>3</intProperty>
      <stringProperty>prop3</stringProperty>
   </uc4ArrayParent>
</uc4TopParent>
```

As with the axample above, XQuery should include root node and array name when parsing. For example, XQuery:

```
for $x in $input/uc4TopParent/uc4ArrayParent
return $x/stringProperty/text()
```

The parsing results will be displayed in the **Result box** as shown below.


Response Expression Tester

Test Input Replace with the response of the last execution.

```
<uc4TopParent>
  <uc4ArrayParent>
    <intProperty>1</intProperty>
    <stringProperty>prop1</stringProperty>
  </uc4ArrayParent>
  <uc4ArrayParent>
    <intProperty>2</intProperty>
    <stringProperty>prop2</stringProperty>
  </uc4ArrayParent>
  <uc4ArrayParent>
    <intProperty>3</intProperty>
    <stringProperty>prop3</stringProperty>
  </uc4ArrayParent>
</uc4TopParent>
```

Parsing Type: XQuery

Expression:

```
for $x in $input/uc4TopParent/uc4ArrayParent
return $x/stringProperty/text()
```

Evaluate

Result: prop1,prop2,prop3

2.5 Managing Job Reports and Optional Reports

Using the Rapid Automation page, you can specify how Job reports are saved, and whether they include Agent log information.

Specifying Job Report Settings

Using the settings in the **Job Report** section, you can specify how Job reports are saved.

- **Store to**
  - **Database:** When checked, the Job report is managed in the Automation Engine. After the execution of a Job, the process protocol is transferred to the Automation Engine database via file transfer.
  - **File:** When checked, the Job report is managed in the Automation Engine. After the execution of a Job, the process protocol is available in the target system as a file.
- **Generate on error only**

  Determines when the Job report is stored in the Automation Engine database and/or a file in the target system. Options are:

  - Always
  - On error only

  This function is only available when **Database** and/or **File** are checked above.

Specifying Whether to Write Agent Log to the Job Report

Using the settings in the **Write agent log, to job report** option in the **Optional reports** section, you can determine whether Job reports will include an Agent log tab with an Agent log when there are errors. Keeping this box checked is recommended for troubleshooting purposes.

2.6 Overriding RA Web Service SOAP Agent Job Attributes

You can read and set the specific attributes of RA Web Service SOAP Agent Jobs by using the script elements :GET_ATT and :PUT_ATT. This topic lists the available attributes for Web Service Jobs.

Job attributes are case-sensitive.

General Settings

Field	XML Name	Available Values
Connection	webConnection	Text
Port	port	Text
Operation	methodName	Text
Include required fields only in generated XML request	requiredOnly	• true • false
Remove template values from generated XML request	removeTemplateValues	• true • false

URL Input and Reporting Settings

Field	XML Name	Available Values
URL Endpoint	urlEndpoint	Text
Write request and response to log	printResponseRequestDebug	• true • false
Create XML SOAP request report	printSOAPRequest	• true • false
Create XML SOAP response report	printSOAPResponse	• true • false

2.7 Setting Trace

When you need to troubleshoot a Rapid Automation Agent loaded into an Automation Engine system, you can turn on Rapid Automation trace for AGENT and JOBS objects from either the System Overview dialog in the Automation Engine user interface or the ucxjcitx.ini file. You can turn on Rapid Automation user interface trace from the uc4config.xml file.

Turning Rapid Automation Trace On or Off From the System Overview Dialog

To turn on trace for Rapid Automation AGENT and JOBS objects in the user interface, go to **System Overview** dialog, right-click the Agent, pick **Properties**. In the dialog that pops up, set **RA** to **99** to turn trace on or set **RA** to **0** to turn trace off.

Turning Rapid Automation Trace On or Off From the ucxjcitx.ini File

To turn on trace for Rapid Automation AGENT and JOBS objects with the **ucxjcitx.ini** file:

1. Stop the Agent(s) from the **ServiceManager** window or command line.
2. Edit the **ucxjcitx.ini** file.

To turn trace:	Set:
On for the Rapid Automation Agent, and its third-party libraries	ra=99
Off	ra=0

   Third-party library trace can only be set in the **ucxjcitx.ini** file. It cannot be set through the UserInterface like for other Agents
3. In the code below, **ra=99** turns Rapid Automation trace on:
   ```
   [TRACE]
   file=..\temp\RA_TRACE_##.TXT
   max_trace_kb=8000
   tcp/ip=0
   ra=99
   trccount=10
   ```
4. Restart the Agent(s).

For more information on setting trace, see your Automation Engine documentation.

Release Highlights for the RA Web Service SOAP Agent v4

No Upgrade from v2 or v3 RA Web Service Agent

There are two separate Web Service Agents v4 – one for SOAP and one for REST objects. There is no automatic upgrade from Web Service v2 or v3 Agent to Web Service v4 Agents. There is a migration assistant for RA Web Service REST Agent Jobs and Connection objects of RA Web Service Agent v3 to RA Web Service REST Agent v4. Furthermore, RA Web Service REST Agent v4 doesn't replace Web Service Agent v2 or v3 solution, it needs to be installed on a separate RA Agent. Consequently, a combination of Web Service v3 and v4 Agents can be run in parallel within one Automation Engine system if those versions are supported on that Automation Engine version. For information on which Rapid Automation Agents are supported on with which Automation Engine versions, see the Automic Compatibility Matrix.

New Features Only in Major and Minor Versions of All Automic Software

Automic recommends that you always install the latest Service Pack or Hotfix. Both contain valuable corrections and bug fixes between Major and Minor Releases, where new features and enhancements are introduced.

- **Major Release**: This is the main version of a software release. It is identified by the first segment of the entire version number (such as the 10 in **10**.0.0).
- **Minor Release:** This includes new features, modifications and corrections that may contain major changes such as database modifications. It is identified by the second segment of the entire version number (such as the 2 in 11.**2**.0).
- **Service Pack:** This is a patch for a release and contains corrections for errors. New features or modifications are not included. Service packs are identified by the third segment of the entire version number (such as the 2 in 10.0.**2**).
- **Hotfix**: This is a minor sub-release to remove malfunctions and defects. Hotfixes are indicated by an HF number after the version number (such as the 1 in 10.0.2 **HF 1**).

Getting the Latest Information

Documentation, release notes, and other information is often updated after software is released. The table below shows where to find the most recent information for Automic software releases.

To find the most recent:	Go to the:
Bug fixes, known issues, and workarounds	Automic Download Center
HTML5 documentation and .pdf files for documentation and release notes	Automic Hosted Documentation
Compatibility for Automic software components, versions, and sub-components	Automic Compatibility Matrix

The Web Service Rapid Automation Agent is not available on IBM AIX operating systems.

Java Requirements

On the host, check the current version of your system's Java Virtual Machine (JVM) using the following command:

```
java -version
```

The Web Service SOAP Agent requires Java JDK 1.7 or 1.8 on the Agent machine.

On the Agent machine, the explicit path to JDK Java 1.7 or 1.8 is needed on the command to start the Agent as shown below:

```
/etc/alternatives/jdk1.7.0_45/bin/java  -Xmx2048m  -jar ucxjcitx.jar
disable_cache
```

For platform-specific Java requirements on your Automation Engine machine, see your Automation Engine release notes.

You can get the necessary files to install Java from Oracle.

v4.1.0

What's New

- Kerberos is added as an **Authentication** option in the RA Web Service SOAP Agent Connection object.
- The new Use Native mode setting in the Agent definition allows you to omit a check for the JDK and work without CXF.
- XOP is added as a transfer type for attachments.

Change in Behavior

WS-Security is no longer supported.

v4.0.0

What's New

- The RA Web Service SOAP Agent has been redesigned and is now supported for the Automic Web Interface, which is available for Automation Engine v12 and above.
- The RA Web Service SOAP Agent v4.0 is based on different libs/solution than v3.x. V4.0 uses CXF for RA Web Service SOAP Agent Jobs, while v3 used AXIS2.