JIRA Installation

Here are the steps to install the solution with JIRA.

Step 1: Turn off JIRA

Step 2: Install the binary

1. Retrieve the rsaaj.jar file from theInstallation page.
2. Retrieve the 3 extra binaries from Go2Group by creating a JIRA Issue athttps://jira.go2group.com/browse/RSAA
3. Install the binary to the WEB-INF/libfolder in JIRA
   a. **Example Location**

      ```
      $JIRA_INSTALLATION/atlassian-jira/WEB-INF/lib
      ```

Step 3: Edit the seraph-config.xml

1. Locate your seraph-config.xml
   a. **Example Location**

      ```
      $JIRA_INSTALLATION/atlassian-jira/WEB-INF/classes/seraph-config.xml
      ```

2. Replace / comment the existing<authenticator> tag and replace it with Go2Group's custom authenticator:
   a. This is usually the default authenticator:

      Example Location

      ```
      <authenticator class="com.atlassian.jira.security.login.JiraSeraphAuthenticator"/>
      ```

   b. Comment it and replace it with:

      Example Location

      ```
      <authenticator class="com.go2group.jira.rsaaj.RSAAJAuthenticator"/>
      ```

Step 4: Setup your license files

In the installation package, Clients should also receive a license file. Your license file should be in the format of **rsaaj.license** and must be installed in this location:

```
$JIRA_INSTALLATION/atlassian-jira/WEB-INF/classes
```

Step 5: Setup your binary configuration file

In the installation package, Client should also find two property files.

rsa-securid.properties

This property file is described below:

rsa-securid.properties

```
### Enable Solution                                                ###
### Values: true,  false                                       ###
rsasecurid.enable=true
### Single or Two Factor                                       ###
### Single factor is authentication with RSA Only        ###
### Two factor is RSA first then Atlassian                        ###
### Values: single, two
rsasecurid.factor=two
### RSA SecurID Token Separator                                    ###
### Must be Java escaped                                       ###
rsasecurid.token.separator=\\|
### RSA SecurID Token Code length                                 ###
rsasecurid.token.code.length=8
```

rsa_api.properties

This is the configuration file for the Solution using RSA's Authentication API to talk to the RSA Authentication Manager.

Important configurations to edit:

Key	Default Value	Recommended Value	Description
RSA_AGENT_HOST	_BLANK_	_BLANK_	This is the IP address for your RSA Authentication Manager.
SDCONF_LOC	_BLANK_	$JIRA_HOME/rsaa/sdconf.rec	Indicates the absolute path to the node secret that will perform the authentication for you RSA Authentication Agent. This file should be provided by the RSA Authentication Manager.
SDSTATUS_LOC	_BLANK_	$JIRA_HOME/rsaa/JAStatus.1	Indicates the absolute path to the Authentication Manager server status file.
SDNDSCRT_LOC	_BLANK_	$JIRA_HOME/rsaa/nodesecret.rec	Indicates the absolute pathto the Authentication Manager node secret file. This file should be provided by the RSA Authentication Manager.

rsa_api.properties

```
#        RSA Authentication API Properties
#        Override Host IP Address
RSA_AGENT_HOST=

#        Interval in seconds between which configuration is refreshed.
RSA_CONFIG_READ_INTERVAL=5

#        [This section is for Data Repository configuration.]
#        Type of the Server configuration.
SDCONF_TYPE=FILE
#        Path of the Server configuration.
SDCONF_LOC=sdconf.rec
#        Type of the Server statuses.
SDSTATUS_TYPE=FILE
#        Path of the Server statuses.
SDSTATUS_LOC=JAStatus.1
#        Type of the Server options.
SDOPTS_TYPE=FILE
#        Path of the Server options.
#SDOPTS_LOC=sdopts.rec
SDOPTS_LOC=
#        Type of the Node Secret.
SDNDSCRT_TYPE=FILE
#        Path of the Node Secret.
#SDNDSCRT_LOC=

#        [This section is for event logger.]
#        Logs event messages to the console.
RSA_LOG_TO_CONSOLE=YES
#        Logs event messages to a file.
RSA_LOG_TO_FILE=NO
#        Name of the log file.
RSA_LOG_FILE=rsa_api.log
#        Minimum severity level allowed to log.
RSA_LOG_LEVEL=DEBUG

#        [This section is for debugger.]
#        Enables debug tracing.
RSA_ENABLE_DEBUG=NO
#        Sends tracing to the console.
RSA_DEBUG_TO_CONSOLE=NO
#        Sends tracing to a file.
RSA_DEBUG_TO_FILE=NO
#        Name of the trace file.
RSA_DEBUG_FILE=rsa_api_debug.log
#        Allows function entry tracing.
RSA_DEBUG_ENTRY=NO
#        Allows function exit tracing.
RSA_DEBUG_EXIT=NO
#        Allows control flow tracing.
RSA_DEBUG_FLOW=NO
#        Allows regular tracing.
RSA_DEBUG_NORMAL=NO
#        Traces the location.
RSA_DEBUG_LOCATION=NO
```

Step 6: Start JIRA and log

After setting up the above, Client's can test their login by entering the username and RSA SecurID token and Atlassian password (depending on the **rsa-securid.properties**). For example, with the following configuration:

- rsasecurid.enable is **true**
- rsasecurid.factor is **two**
- rsasecurid.token.separator is **\\|**

The user needs to login with the username of <**username**> and password equaling to <**securid_token**>**|**<**atlassian_password**>.

- Username: **rsaajtester**
- Password:**84342143|rsaajpass**