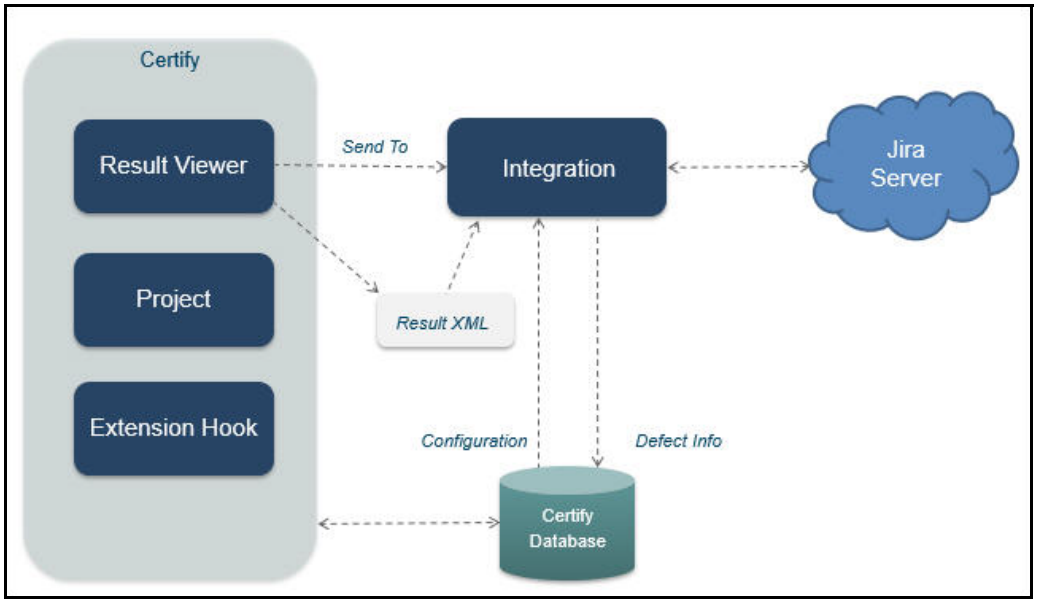
Integrating Worksoft Certify with Atlassian Jira

Worksoft Certify® integration with Atlassian® Jira® enables users to submit issues into Jira from the Certify Result Viewer. Jira is an IT issue management system that allows users to capture and organize IT issues, assign work, and follow team activities.

With the integration software, you can create Jira issues and populate all of the issue fields using information from Certify results and processes. The issue is then imported into a Jira system, where you can edit and refine it. After the issue is submitted to Jira, you cannot see the details of the issue in Certify.

Integration Architecture

The following diagram shows an overview of how this integration works.



In the Certify Result Viewer, a user right-clicks on a failed step and selects **Send To > Jira**. Certify generates a result XML file and an image file that is sent to the Jira Integration tool.

The Integration tool completes the following tasks:

- Reads the result file and image file
- Generates a Jira issue
- Populates fields for the issue in Jira by using Certify process and results data
- Submits the issue to Jira which provides back an ID for a new issue
- Records the issue ID into the Certify database
- Launches Jira to allow users to do additional editing on the new issue

Jira Integration Command Line Modes

The Jira Integration tool is implemented using a standalone command line executable. This executable has the following modes of operation:

- Help
- GET_METADATA
- SET_CONFIG
- GET_CONFIG
- PROJECT
- CREATE_TEMPLATE
- REQUEST_OAUTH
- LIST_PROJECTS

When you start the Jira Integration executable without any arguments, you will see a summary of all the modes.

```
C:\Program Files (x86)\Worksoft\Certify\Client>jiraintegration.exe
JiraIntegration.exe can be called 5 ways:

* JiraIntegration.exe
  - Called with no arguments in order to see this help message

* JiraIntegration.exe GET_METADATA filename
  - Extract metadata from JIRA and write to the file specified

* JiraIntegration.exe SET_CONFIG filename
  - Save configuration data from the file specified to Certify database

* JiraIntegration.exe GET_CONFIG filename
  - Get configuration data from Certify database to the file specified

* JiraIntegration.exe PROJECT projectname filename
  - Extract result details from file then populate and submit an issue to JIRA for the specified proje

Press enter key to continue.
```

Help Mode

The Jira Integration executable, JiraIntegration.exe, starts in the Help mode. In this mode, Jira configuration stored in Certify is retrieved, and an attempt will be made to connect with the Jira system. Any conditions resulting in an error will be shown to the user.

GET_METADATA Mode

Use the GET_METADATA mode to extract metadata from Jira and write it to a specified file. This mode is useful for retrieving details about fields from the back-end system, and it will most likely be used by someone editing fields in configuration data.

Before performing this operation, you must have saved the Jira URL, user ID, and password in Certify through option SET_CONFIG. Certify uses these Jira credentials to establish a connection with the Jira System and retrieve meta data.

SET_CONFIG Mode

The SET_CONFIG mode stores the configuration data from a specified XML file to a Certify database. The two authentication methods are supported by Jira integration:

- Basic Auth
- OAuth

Basic Auth Method

To use the Basic Auth method, you must use an API token. API tokens are created and managed through the user's Atlassian account settings:

https://support.atlassian.com/atlassian-account/docs/manage-api-tokens-for-your-atlassian-account/

OAuth Method

To use the OAuth method, see the "Configuring the Jira Connection" on page 11.

Example 1:

```
<DETAILS>
    <Credentials>
        <AuthenticationMode>BASIC</AuthenticationMode>
        <USERNAME>myuserid</USERNAME>
        <PASSWORD>letmein</PASSWORD>
        <ConsumerKey/>
        <Verifier/>
        <AccessToken/>
        <TokenSecret/>
        <PrivateKey/>
    </Credentials>
</DETAILS>
```

This configuration saves Jira credentials into Certify. You must have at least this configuration saved in Certify before performing the GET_METADATA operation.

Example 2:

```
<DETAILS>
   <MyProjectName>
      <FIELD>
         <FIELDSTRING>"project":{"key":"CER"}</FIELDSTRING>
      </FIELD>
      <FIELD>
         <FIELDSTRING>"summary":"{0}"</FIELDSTRING>
         <VALUE_PATH>CertifyResults/LogTestStep/LogTestStepDetails/Narrative</VALUE_PATH>
      </FIELD>
      <FIELD>
         <FIELDSTRING>"issuetype":{"name":"{0}"}</FIELDSTRING>
         <DEFAULT_VALUE>Bug</DEFAULT_VALUE>
      </FIELD>
      <FIELD>
         <FIELDSTRING>"reporter":{"name":"{0}"}</FIELDSTRING>
         <DEFAULT_VALUE>CertifyTester1</DEFAULT_VALUE>
         <FIELD_NAME>Reporter</FIELD_NAME>
      </FIELD>
      <FIELD>
         <FIELDSTRING>"description":"{0}"</FIELDSTRING>
         <VALUE_PATH>CertifyResults/LogTestStep/LogTestStepDetails/ErrorMsg</VALUE_PATH>
         <FIELD_NAME>Description</FIELD_NAME>
      </FIELD>
      <FIELD>
         <FIELDSTRING>"assignee":{"name":"{0}"}</FIELDSTRING>
         <DEFAULT_VALUE>CertifyTester1</DEFAULT_VALUE>
         <FIELD_NAME>Assignee</FIELD_NAME>
      </FIELD>
   </MyProjectName>
      <Credentials>
         <AuthenticationMode>BASIC</AuthenticationMode>
         <USERNAME>myuserid</USERNAME>
         <PASSWORD>letmein</PASSWORD>
      </Credentials>
      <URI>https://worksoftdefect.atlassian.net/rest/api/2/issue/</URI>
</DETAILS>
```

Example 3:

```
<DETAILS>
   <MyProjectName>
      <FIELD>
         <FIELDSTRING>"project":{"key":"CER"}</FIELDSTRING>
      </FIELD>
      <FIELD>
         <FIELDSTRING>"summary":"{0}"</FIELDSTRING>
         <PROCESSATTRIBUTE>AttributeName</PROCESSATTRIBUTE>
      </FIELD>
      <FIELD>
         <FIELDSTRING>"issuetype":{"name":"{0}"}</FIELDSTRING>
         <DEFAULT_VALUE>Bug</DEFAULT_VALUE>
      </FIELD>
      <FIELD>
         <FIELDSTRING>"reporter":{"name":"{0}"}</FIELDSTRING>
         <VALUE_PATH>CertifyResults/LogTestStep/LogTestStepParms/Param[@Name='Returned
         Value']</VALUE_PATH>
      </FIELD>
      <FIELD>
         <FIELDSTRING>"description":"{0}"</FIELDSTRING>
         <PROCESSDESCRIPTION></PROCESSDESCRIPTION>
      </FIELD>
      <FIELD>
         <FIELDSTRING>"assignee":{"name":"{0}"}</FIELDSTRING>
         <DEFAULT_VALUE>CertifyTester1</DEFAULT_VALUE>
      </FIELD>
   </MyProjectName>
      <Credentials>
         <AuthenticationMode>BASIC</AuthenticationMode>
         <USERNAME>myuserid</USERNAME>
         <PASSWORD>letmein</PASSWORD>
      </Credentials>
      <URI>https://worksoftdefect.atlassian.net/rest/api/2/issue/</URI>
</DETAILS>
```

The password field is specified in clear text but is encrypted when stored in the Certify database. Fields specify which Jira fields to submit as part of the ticket.

The project name is a child of the root node field <DETAILS>. This project name should reflect your exact name in Certify when invoking Jira integration execution.

Each configuration is identified by the project name that surround all <FIELD> tags. Certify can hold multiple configurations where each configuration is identified by unique project name.

- Value can be hard-coded in the case of the project field.

  Example: `<FIELDSTRING>"project":{"key":"JI"}</FIELDSTRING>`

- Value can be extracted from a Certify result

  Example: `<FIELDSTRING>"summary":"{0}"</FIELDSTRING>`

  where {0} is the token to be replaced by a value

  Example: `<VALUEPATH>CertifyResults/LogTestStep/LogTestStepDetails/Narrative</VALUEPATH>`

  where value path is the XPath expression for extracting a value from a Certify result XML

- Value can be extracted from a process attribute

  Example: `<FIELDSTRING>"description2":"Temp {0} description"</FIELDSTRING>`

  where {0} is the token to be replaced by a value

  Example: `<PROCESSATTRIBUTE>AttributeName</PROCESSATTRIBUTE>`

  where attribute name is the name of the process attribute from which to select the value

- Value can be extracted from a process description

  Example: `<FIELDSTRING>>"description4":"Temp {0} description"</FIELDSTRING>`

  where {0} is the token to be replaced by a value

  `<PROCESSDESCRIPTION></PROCESSDESCRIPTION>`

  tells the integration to use the description of the top level process

- Value can be extracted from a linked requirement name

  `<FIELDSTRING>>"description3":"Temp {0} description"</FIELDSTRING>`

  where {0} is the token to be replaced by a value

  `<FIRSTLINKEDREQUIREMENT></FIRSTLINKEDREQUIREMENT>`

  tells the integration to find the first linked requirement and to extract its name

- Value can be extracted from a value string.

  Example:

  ```
  <FIELD>
     <FIELDSTRING>"summary":"{0}"</FIELDSTRING>
     <VALUE_STRING>{{STEP_ERROR_MSG}}</VALUE_PATH>
     <FIELD_NAME>Summary</FIELD_NAME>
  </FIELD>
  ```

  The <VALUE_STRING> tag specifies a template string that contains one or more token expressions. A token expression takes the form of {{TOKEN}}. When an interpolated string is evaluated, the token expressions are replaced by the string representations of the expression result.

There are four predefined tokens:

- {{STEP_WINDOW_NAME}}
- {{STEP_OBJECT_NAME}}
- {{STEP_ID}}
- {{STEP_ERROR_MSG}}

A token expression can be an XPath query for selecting a value from a Certify results XML similar to the <VALUEPATH> tag.

The following example shows how multiple result values can be combined into a single string:

<VALUE_STRING>window name: `{{STEP_WINDOW_NAME}}`

object name: `{{STEP_OBJECT_NAME}}`

step Id: `{{STEP_ID}}`

error message: {`{{STEP_ERROR_MSG}}`

log header: `{{/CertifyResults/LogTestStep/LogTestStepDetails/LogHeader}}</VALUE_STRING>`

which will produce a result similar to the following:

window name: `BCALV_GRID_01:0100`

object name: `GridView (Flights)`

step Id: `1512`

error message: `GuiCtrlGridViewFindRow: Value(s) not found. The given grid contains 100 number of rows. The very first visible row in gridview is at 94 position.`

log header: `SAP End to End_New - 6/21/2022 9:28:19 AM`

GET_CONFIG Mode

The GET_CONFIG mode extracts configuration details from the Certify database and stores the details in a specified file.

The password will not be decrypted to prevent anyone from knowing the Jira password. If the mapping file is updated and needs to be saved back to Certify, you can do one of the following to protect your password:

- Omit the password tag.

  Password field in the database will not be updated.
- Provide the correct clear-text value for the password.

The GET_CONFIG mode also has an optional project name argument to retrieve the configuration for a specified project.

Example: `JiraIntegration.exe project filename`

If no project name is specified, then the configuration for the default project is retrieved. The default project is the first project that was saved in SET_CONFIG. You can not change the default project.

PROJECT Mode

The PROJECT mode extracts result details from the Certify Result file, then populates and submits an issue to Jira for a specified project.

You will need to provide the following two parameters for the PROJECT mode:

- Projectname - Refers to project name in the Jira system where the Jira issue is created. The project name corresponds to same project name that is defined in the configuration.
- Filename - Refers to the Certify Result file that is supplied by Certify when the PROJECT mode is invoked from Certify.

The Project option is intended to be used only from Certify where the file name is automatically provided by Certify during the execution.

PROJECT Issue Mode

The PROJECT Issue mode extracts result details from the result file, queries the Certify database for details of the target Jira system, and generates a Jira issue. If the integration was successful, then details of the Jira issue will be associated with the result in the Certify database.

An entry will be created in a table called External Issue, and the entity ID field will be the ID of the result log header. Details will be added to a child table called External Issue Details, and these details will include the ID and key of the Jira issue. If it is not successful, you will receive an error message. If you have an image attachment for the result step, it will be attached to the Jira issue.

CREATE_TEMPLATE Mode

The CREATE_TEMPLATE mode creates a new empty configuration file that can be used as a starting point for configuring Jira integration.

You will need to provide the following parameter for the CREATE_TEMPLATE mode:

- Filename - Refers to the configuration file where the Jira template configuration file should be saved.

REQUEST_OAUTH Mode

The REQUEST_OAUTH mode uses the credentials configuration details from either the Certify database, if previously set, or a specified file. This mode performs the necessary steps to request an access token from Jira, and then stores the data to the specified file.

This mode requires that the `<ConsumerKey/>` and `<PrivateKey/>` are set.

You will need to provide the following parameter for the REQUEST_OAUTH mode:

- Filename - Refers to the configuration file where the Jira OAuth configuration data should be saved.

Example:

```
<DETAILS>
    <Credentials>
        <AuthenticationMode>OAUTH</AuthenticationMode>
        <ConsumerKey>MyApplicationName</ConsumerKey>
        <PrivateKey>-----BEGIN PRIVATE KEY-----
            MIIcJIntrFr3BGkQhKIg9W0baQefAAsCAMEwGgJDAgEAAOgbankoExPafiYGLm+D
            rxscuTKZFMq9zgQW30DbLgbGemQCB98bLn/xBeZ88gQwLRKpw4+nlXO2nYbn2vvL
            k/INnHtrwCgL/pL9L72SQm53SDnnT+CU8chJirz5536NaiFdFfkzE50HDxFESqWX
            ICLDWckDvahyx7jhzyBvGshsMFcFAgmbaaeCgyBzR9KqivUVTQF+3djoG055fOiB
            Xg0koJU+F10AvcbLzMQVvXg1w9c9B6fw9dHFFXWLILRLHdFud6s5tlYazPb9T7gY
            jQlITKyaUgwsf9IaSyoGz0ELimmz6XHjTcRFY7yWTeKcIFoEQ3trIfnyATtHDNZY
            53+XnE95ssdh/hgoIQjBApsqptKaBn6JBniXzao45CzlqmTUUxjzWkdYNQU1auZA
            2qe5xPANyRA8L0TrQjG9puQIn4k87WvXNdSmdZWe/WKcQqdJT8DY6eSKXUfGNSA2
            bMyCD8EKCogalmsfb/HkqKQo8pJVV4hQLu+1nm5aWbucCt0ZT8P2D/fdMHwQnFzz
            SJsdAkeatg2GJ3D4ICO+4HmG8dB7vaSlwIsbFLvFs6EwEASjK4T9sEDeSKIoNduS
            AS02puqXqa8SwFHKn846C9PuQMAOwQJBAM0vuKQa15JTXnx8mRVHxJsqyV/c/dqy
            ICAgV29ya3NvZnQgQ2VydGlmeSBKaXJhIEludGVncmF0aW9uIHdpdGggT0F1dGgg
            LS0tPiBSYWxlaWdoIEZyYW5rbGluIFJpbmVoYXJ0IElJSSA8LS0tLi4uLi4xMjMh
            1aBcDeFGhIjKlMn=
            -----END PRIVATE KEY-----
        </PrivateKey>
    </Credentials>
</DETAILS>
```

Preparing Integration with Jira

For Certify to integrate with Jira, the following tasks must be performed:

- Create an Extension hook in Certify
- Configure Jira
- Map Jira fields

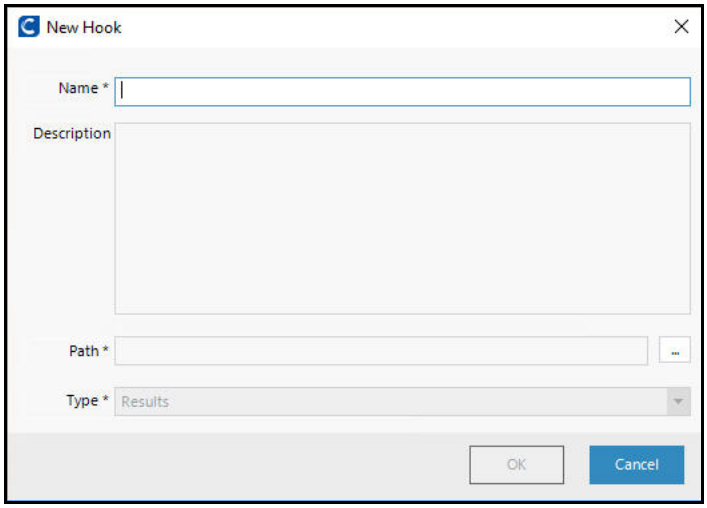
Creating an Extension Hook in Certify

The Jira integration is implemented by creating an Extension hook in Certify and associating the hook to a Certify project.

▶ *To create a hook:*

1. In the Certify Navigation pane, click **Extensions**.
2. In the Navigation tree, select **Extensions > Hooks > Result**.
3. Right-click in the Summary pane and select **New Hook**.

   The New Hook dialog opens.



4. In the Name field, enter **`Jira`**.
5. In the Description field, enter **`Jira Hook`**.
6. In the Path field, enter the path where the **JiraIntegration.exe** file is found in the Worksoft Certify client folder:

   ```
   . . .Worksoft\Certify\Client\JiraIntegration.exe PROJECT myprojectName
   ```

   The keyword PROJECT must be followed by the same project name that was used in the configuration earlier. Certify cannot create a Jira ticket if it is unable to find the project name.
7. Click **OK**.

▶ *To add a hook to a project:*

1. In the Certify Navigation pane, click **Projects**.

   The Projects window opens.
2. In the Summary pane, select a project.
3. Click the **Hooks** tab.
4. Right-click in the Hooks tab and select **Add Hook**.

   The Select Hooks dialog opens.
5. Select a hook in the Summary pane.
6. Click **OK**.

   The hook is added to the project and appears in the Hooks tab. Also, the Jira option appears in the right-click menu option **Send To** in the Results Viewer Summary pane.

Configuring the Jira Connection

For Certify to integrate with Jira, the following tasks must be performed:

- Create and populate a configuration file for the Jira Integration executable. The file requires a URI and Jira credentials data.

  This information gets stored in the Certify database along with details of which fields to populate for a Jira issue.
- Save the configuration file to the Certify database.

Use the procedure appropriate for your selected authentication method:

- Basic Auth
- OAuth

Creating a Configuration File for Basic Auth

For Basic Auth, you must use the CREATE_TEMPLATE mode to create the configuration file.

To use the Basic Auth method, you must use an API token. API tokens are created and managed through the user's Atlassian account settings:

https://support.atlassian.com/atlassian-account/docs/manage-api-tokens-for-your-atlassian-account/

▶ *To create a configuration file for Basic Auth:*

1. In a text editor, use the CREATE_TEMPLATE mode to create a file with the following code:

```
<Credentials>
    <AuthenticationMode>BASIC</AuthenticationMode>
    <USERNAME>myuserid</USERNAME>
    <PASSWORD>letmein</PASSWORD>
</Credentials>
<URI>https://worksoftdefect.atlassian.net/rest/api/2/issue/</URI>
```

2 Add the following Jira information:

- Username
- Password
- URI

3 Save the updated configuration file as an XML file.

Example:

```
. . . C:\Temp\JiraTemplate.xml
```

You now need to save the configuration file to the Certify database. For more information, see "Saving the Configuration File to the Certify Database" on page 14.

Create a Configuration File for OAuth

To use OAuth with Jira, you must create an RSA public/private key pair using OpenSSL. Windows must also be installed on Mac and Linux machines.

After you have created your key pair, you must make an application link in Jira. After creating this link, you will generate your configuration file.

▶ *To generate an RSA public/private key pair:*

The files generated are saved in the directory where you run the commands.

1 Open the Command Prompt window and run the following OpenSSL command:

```
openssl genrsa -out certify_privatekey.pem 1024
```

2 Create an x509 certificate using the private key.

```
openssl req -newkey rsa:1024 -x509 -key certify_privatekey.pem -out
certif_publickey.cer -days 365
```

3 Extract the public key from the certificate.

```
openssl x509 -pubkey -noout -in certif_publickey.cer  > certify_publickey.pem
```

4 Convert the private key to a PKCS8 format.

```
openssl pkcs8 -topk8 -nocrypt -in certify_privatekey.pem -out
certify_privatekey.pcks8
```

The following key pair files are created:

- `certify_privatekey.pem`
- `certif_publickey.cer`
- `certify_publickey.pem`
- `certify_privatekey.pcks8`

You will now create an application link in Jira.

▶ *To create an application link in Jira:*

1. Open Jira and select **Jira settings** (cog icon) **> Applications**.
2. In the left sidebar, select **Application Links**.
3. In the Enter the URL of the application field, enter the URL you want to link.

   Example: `http://example.com/`
4. Click **Create new link**.

   The Link applications dialog opens.

   If you get a warning that says, "No response was received from the URL you entered," ignore it.
5. In the Link applications dialog, select the **Create incoming link** option.

   Since you just want to retrieve data from Jira, you do not have to enter information in the remaining fields.
6. Click **Continue**.
7. In the next Link applications screen, type in the following consumer details for the Certify Jira Integration client:
   - Consumer key: **CertifyOauthKey**
   - Consumer name: **Certify Jira Integration**
   - Public key: Paste the public key from the **`certify_publickey.pem`** that was previously generated.
8. Click **Continue**.

   Your application link in Jira is now completed, and you are ready to create the configuration file.

▶ *To create a configuration file for OAuth:*

1. In the Command Prompt window, run **JiraIntegration.exe** and use the **CREATE_TEMPLATE** command, specifying the configuration file name where the template should be saved.
2. Open the template file in a text editor like Notepad.
3. Modify the template file by setting the **ConsumerKey** to **`CertifyOauthKey`** and **PrivateKey** to the value from the **`certify_privatekey.pcks8`** file.

   Example:

```
<ConsumerKey>CertifyOauthKey</ConsumerKey>
        <PrivateKey>-----BEGIN PRIVATE KEY-----
            MIIcJIntrFr3BGkQhKIg9W0baQefAAsCAMEwGgJDAgEAAOgbankoExPafiYGLm+D
            rxscuTKZFMq9zgQW30DbLgbGemQCB98bLn/xBeZ88gQwLRKpw4+nlXO2nYbn2vvL
            k/INnHtrwCgL/pL9L72SQm53SDnnT+CU8chJirz5536NaiFdFfkzE50HDxFESqWX
           1aBcDeFGhIjKlMn=
            -----END PRIVATE KEY-----
```

4. Set the URI value to the address of the Jira Instance and add "**`/rest/api/2/issue/`**".

   Example: `http://jira.example.com/rest/api/2/issue/`
5. Save the updated configuration file.

6 In the Command Prompt window, run **JiraIntegration.exe** and use the **REQUEST_OAUTH** command to generate the required tokens for OAuth.

Example workflow

```
c:\Program Files (x86)\Worksoft\Certify\Client> JiraIntegration.exe REQUEST_OAUTH jira_config.xml
Request token successful.
Token:  4cq0DBQkLIY8m6JqpxFfADKW8vvcweiQ
Secret: RG91ZywgT2JhaWQsIFJhbGVpZ2gtLS0=
Please open http://joauth.worksoft.com:8080/plugins/servlet/oauth/authorize?oauth_token=4cq0DBQkLIY8m6JqpxFfADKW8vvcweiQ to authorize it.

Enter verification code from Jira (ctrl-c to abort)...
> J6KxTT

Access_Token request successful.
Token:  cmFsZWlnaCBkaWQgdGhpcyEgICAgICAu

c:\Program Files (x86)\Worksoft\Certify\Client>
```

7 When prompted, copy the URL address from the Command Prompt window and paste it into a browser.

8 Press **<Enter>**.

9 Click the **Allow** button in the browser to authorize the request token.

10 Copy the verification code from the browser window and paste the code in the Command Prompt window.

11 Press **<Enter>**.

A message appears stating that the access token request was successful.

```
Access_Token request succcessful.
Token:  j232g6Ndgl2gYRk80clk5qS0IAR0g0a2
```

The next step is to save the configuration file to the Certify database.

Saving the Configuration File to the Certify Database

After you have created your configuration file, you must save it to the Certify database.

▶ *To save the configuration file to the Certify database:*

1 In the Command Prompt window, run **JiraIntegration.exe** in **SET_CONFIG** mode to upload the configuration file to the Certify database.

Example:

```
...Worksoft\Certify\Client\JiraIntegration.exe SET_CONFIG c:\temp\JiraTemplate.xml
```

2 Execute **JiraIntegration.exe** again in **GET_METADATA** mode to extract metadata from Jira and write to a specified file.

Example:

```
...Worksoft\Certify\Client\JiraIntegration.exe GET_METADATA c:\temp\JiraTemplate.xml
```

The remote Jira system is contacted, extracts metadata, and stores the data in the specified file.

3 After you have added the metadata, execute **JiraIntegration.exe** in **SET_CONFIG** mode again to upload the updated configuration file to the Certify database.

Example:

```
...Worksoft\Certify\Client\JiraIntegration.exe SET_CONFIG c:\temp\JiraTemplate.xml
```

4 *(Optional)* If you want to retrieve the Jira configuration now stored in Certify, run the Jira executable in **GET_CONFIG** mode to retrieve the configuration.

Example:

```
...Worksoft\Certify\Client\JiraIntegration.exe GET_CONFIG c:\temp\JiraTemplate.xml
```

Mapping Jira Fields

When an issue is created and submitted into Jira, it must have values in the required fields. You must identify required fields, optional fields, and values by consulting your Jira administrator.

You can also run the Jira Integration executable in **GET_METADATA** mode to identify needed fields. The output file includes details about projects, fields, and values from your Jira instance.

Building Mapping Configuration

Mapping information is specified in an XML file that is subsequently loaded into the Certify database. This file has the following structure:

```
<DETAILS>
    <MyProjectName>
        <FIELDS>
            <FIELD>Field 1 details</FIELD>
            <FIELD>Field 2 details</FIELD>
            <FIELD>Field 3 details</FIELD>
        </FIELDS>
    </MyProjectName>
    <Credentials>
        <AuthenticationMode>BASIC</AuthenticationMode>
        <USERNAME>myuserid</USERNAME>
        <PASSWORD>letmein</PASSWORD>
    </Credentials>
    <URI>https://worksoftdefect.atlassian.net/rest/api/2/issue/</URI>
</DETAILS>
```

Username and password fields are optional. Theses fields need to be loaded into Certify one time, and they do not need to be loaded with field definitions.

Where to Get Data for Jira Fields

As part of your implementation, you need to determine where to get data for the fields in your Jira issue. Data can be gathered from the following places:

- Hard-coded value - A value is hard-coded in the configuration

  ```
  <FIELD>
  <FIELDSTRING>"project":{"key":"JI"}</FIELDSTRING>
  </FIELD>
  ```

  In this case, the project will be identified using a key and will always get a value of JI.

- Result XPATH - A value is retrieved from the result using an XPATH expression.

  ```
  <FIELD>
  <FIELDSTRING>"summary":"{0}"</FIELDSTRING>  <VALUEPATH>CertifyResults/
  LogTestStep/LogTestStepDetails/Narrative</VALUEPATH>
  </FIELD>
  ```

  In this case, the Summary field has a token {0}, which will be replaced by the value extracted by the XPATH expression on the result XML file.

- Process Description - A value is retrieved from the description property of the top-level process in the execution.

  ```
  <FIELD>
  <FIELDSTRING>"description":"{0}"</FIELDSTRING>
  <PROCESSDESCRIPTION></PROCESSDESCRIPTION>
  </FIELD>
  ```

  In this case, the Description field has a token {0}, which the description of the top-level process in the execution will replace.

- Process Attribute - A value is retrieved from a process attribute for the top-level process in the execution.

  ```
  <FIELD>
  <FIELDSTRING>"integrationtype":"{0}"</FIELDSTRING>
  <PROCESSATTRIBUTE>Scope</PROCESSATTRIBUTE>
  </FIELD>
  ```

  In this case, the Integrationtype field has a token {0}, which the value of the Scope attribute of the top-level process in the execution will replace.

- First Linked Requirement - A value is the name of a first linked requirement from a top-level process.

  ```
  <FIELD>
  <FIELDSTRING>"description3":"Temp {0} description"</FIELDSTRING>
  <FIRSTLINKEDREQUIREMENT></FIRSTLINKEDREQUIREMENT>
  </FIELD>
  ```

  In this case, the Description3 field has some fixed text and a token {0}, which the name of the first linked requirement will replace.

- Value String - Values extracted from Certify result data using predefined tokens and/or multiple XPath expressions

```
<FIELD>
  <FIELDSTRING>"summary":"{0}"</FIELDSTRING>
  <VALUE_STRING>Window name: {{STEP_WINDOW_NAME}}.  Error message:
  {{STEP_ERROR_MSG}}</VALUE_STRING>
  <FIELD_NAME>Summary</FIELD_NAME>
</FIELD>
```

  In this case, the Summary field has the token {0} that will be replaced by the value(s) using the predefined tokens(s) and/or XPath expression results from the result XML file as specified by the <VALUE_STRING> tag.

  Replacement values are specified using double brackets ({{). For example, the replacement value's token expression takes the form of {{TOKEN}}. TOKEN is the name of a predefined token or a valid XPath query. Multiple replacement value tokens are allowed because they are hard-coded text.

  There are four predefined tokens:

  - {{STEP_WINDOW_NAME}}
  - {{STEP_OBJECT_NAME}}
  - {{STEP_ID}}
  - {{STEP_ERROR_MSG}}

  The XPath query for selecting a value from the Certify results XML behaves identically to the <VALUEPATH> tag. When specifying an XPath query, the query must begin with a forward-slash (/) character.

Save Field Mapping File to Certify

In the Command Prompt window, execute **JiraIntegration.exe** to load your mapping file into the Certify database.

Next, execute the Jira integration executable in **SET_CONFIG** mode to load your mapping file to the Certify database.

Example:

```
JiraIntegration.exe C:\temp\configuration.xml SET_CONFIG
```

If there are errors, the Command Prompt window indicates the source of the problem.

Submitting an Issue to Jira

After configuring the Jira configuration, you can submit an issue to Jira. You will use the Certify Result Viewer to submit your issues into Jira.

▶ *To submit an issue:*

1 In the Certify Result Viewer, select a process in the Navigation pane.

The Summary pane lists the steps of the process.

2 Right-click on a step and select **Send To > Jira**.

Certify generates a result XML file and an image file that is sent to the Jira Integration tool.

The Jira Integration tool completes the following tasks:

- Reads the result file.
- Reads the image file if one exists.
- Generates a Jira issue.
- Populates fields for the issue in Jira by using a Certify process and results data.
- Submits the issue to Jira which provides back an ID for a new issue.
- Records the issue ID into the Certify database.
- Launches Jira to allow users to do additional editing on the new issue.

While the issue is being constructed and submitted, the Command Prompt window shows the details of the progression.