QADirector

Integration and SDK Reference

Technical support is available from our Technical Support Hotline or via our FrontLine Support Web site.

Technical Support Hotline:
1-800-538-7822
FrontLine Support Web Site:
http://frontline.compuware.com

QADirector: Integration and SDK Reference

October 16, 2008

Revision LOCAL-200810161537

Contents

CHAPTER 1

Introduction

Intended Usage

This Software Development Kit, including the documentation, sample code and APIs provided within, are intended for the express purpose of developing integrations that interface with QADirector. Compuware will provide support for the QADirector application and associated APIs. It is the responsibility of the user to properly utilize the SDK to develop, to debug, to deploy, and to support any applications derived from its usage. It is recommended that any applications developed with this SDK be thoroughly tested in a non-production environment and all data backed up before deploying to a production environment.

NOTE

Only those QADirector API classes referenced within this document are supported.

How to Use This Reference

This reference document includes information about how to use the QADirector SDK/API and integration components. Specifically, it covers:

- Using the QADirector SDK/API defect integration to connect your defect tool to QADirector
- Using the QADirector SDK/API automated testing integration to integrate your automated testing tool with QADirector.
- Using the QADirector SDK/API requirements management integration to integrate your requirements management tool with QADirector.

Related Publications

The QADirector documentation set includes the following:

- The *QADirector Installation Guide* includes system requirements and instructions for installing QADirector. This guide is provided in PDF format.
- The *QADirector Online Help* provides descriptions of the QADirector centers, tools, procedures, and reference information.
- *Distributed License Management Installation Guide* provides instructions for installing and configuring a license for QADirector. This guide is provided in PDF format.

Getting Help

At Compuware, we strive to make our products and documentation the best in the industry. Feedback from our customers helps us maintain our quality standards. Go to `http://www.compuware.com` to access Compuware Corporation's site online. The Compuware site provides a variety of product and support information.

Go to `http://frontline.compuware.com` to access online customer support for Compuware products via the FrontLine support web site. FrontLine provides fast access to critical information about your Compuware products. You can read or download documentation, frequently asked questions, and product fixes, or e-mail your questions or comments. In the **Log In** section, enter your login ID and password and click **Login.** If you are a first-time visitor, click **Register** to receive your free password to access FrontLine. After completing the registration form, your login ID and password are e-mailed to you and your account is activated.

If you need support services, please obtain the following information before calling Compuware's 24-hour product support hotline:

- The name, release (version), and build number of the product. The name and release are on the covers of the product documentation.
- Installation information, including installed options, whether the product uses local or network databases, whether it is installed in the default directories, whether it is a standalone or network installation, and whether it is a client or server installation.
- Environment information, such as the operating system and release on which the product is installed, memory, hardware/network specifications, and the names and releases of other applications that were running.
- The location of the problem in the product software, and the actions taken before the problem occurred.
- The exact product error message, if any.
- The exact application, licensing, or operating system error messages, if any.
- Your Compuware client, office, or site number, if available.

```
Compuware Customer Support
Compuware Corporation
One Campus Martius
```

```
Detroit, MI 48226-5099
Toll free: 800-538-7822
```

```
http://frontline.compuware.com
```

CHAPTER 2

Defect Integration

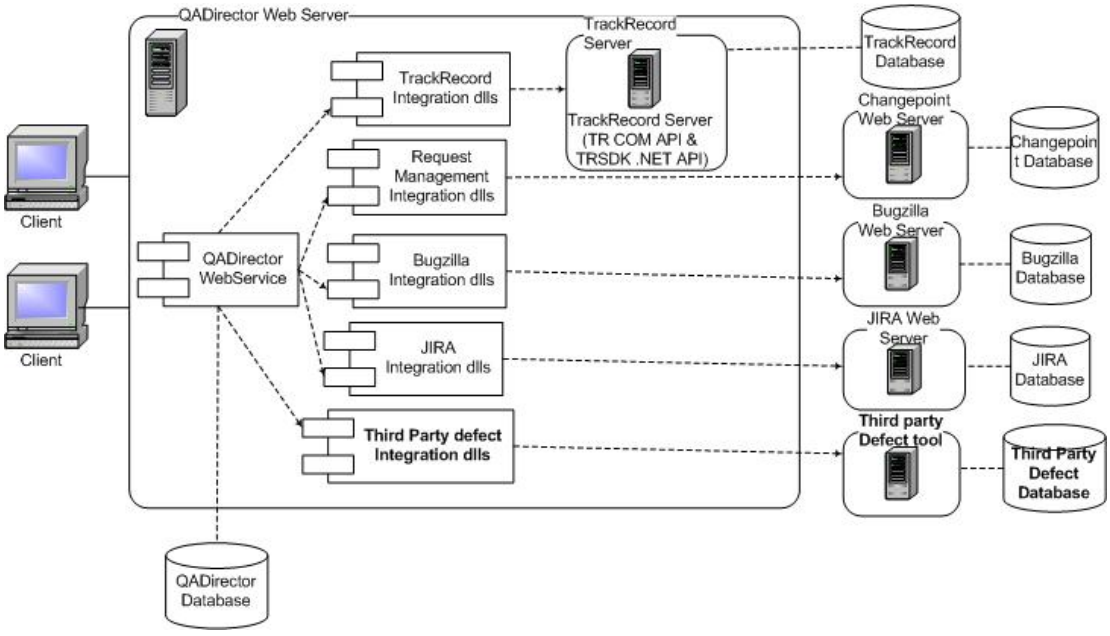
Defect tracking is an integral part of any test management cycle. QADirector facilitates this by providing an open integration for defect tracking tools. When a QADirector test fails, if the defect integration is set up, users can submit defects directly from the failed result in QADirector to their defect tool. The defects submitted from QADirector can be tracked and managed from QADirector's **Defect Center**.

Follow the steps in this section to use any third party defect tool with QADirector. See Deployment [p. 51] after your code is compiled to complete the necessary steps for your integration.

The points of integration include:

- Testing your connection.
- Retrieving defect fields.
- Retrieving defects for display in QADirector's **Defect Center.**
- Submitting defects.
- Editing defects.
- Launching your defect tool from QADirector.

The following illustration shows the QADirector defect tracking integration infrastructure.



System Requirements

- A defect tracking tool.
- Visual Studio 2005 and the .NET framework 2.0 for building your integration DLL.
- Refer to the defect integration C# code samples installed with this SDK for a better understanding of the integration architecture.

Code Reference

For the QADirector defect tracking integration, you will be creating a custom DLL to integrate QADirector with your defect tool. You will need to analyze the necessary APIs and integration points of the defect tool that will integrate with QADirector. QADirector requires that the DLL you create have classes named `IDefectTrackingIntegration` and `ToolClass` which follow the specific format documented in this section. When your DLL is complete, continue to the Deployment [p. 51] section.

IDefectTrackingIntegration Interface

The IDefectTrackingIntegration Interface declares the methods that need to be implemented in the integration application.

See ToolClass [p. 13] for an example of how to create and implement this interface.

ToolClass

`ToolClass` is derived from the `IDefectTrackingIntegration` interface. All of its interface methods should be implemented in this class:

```
interface IDefectTrackingIntegration
{
      string TestConnection(string inputXML);
      string SubmitDefectToTool(string inputXML);
      string GetDefectFieldListFromTool(string inputXML);
      string GetDefectListFromTool(string inputXML);
      string LaunchDefectTool(string inputXML);
      string EditDefectItem(string inputXML);
}

ToolClass:IDefectTrackingIntegration
{
      string TestConnection(string inputXML)
      { //implementation}

      string SubmitDefectToTool(string inputXML)
      { //implementation}

      string GetDefectFieldListFromTool(string inputXML)
      {//implementation}

      string GetDefectListFromTool(string inputXML)
      { //implementation}

      string LaunchDefectTool(string inputXML)
      { //implementation}

      string EditDefectItem(string inputXML)
      { //implementation}
}
```

Parameter Information

The parameters (`string inputXML`) send the integration parameters and their values as defined in the tool and tool domain. Depending on which method is called, the inputXML may contain other information such as fields to include when submitting a defect.

Return Value Information

- In order for QADirector to process the returned data, the return XML string *must* contain the values or error message produced by your implementation of the integration.
- All element names in the return XML must be exactly as the appear in the examples in this section. For example, the element `returncode` cannot be `ReturnCode`.
- If the element `returncode` has `value="0"`, use the `message` element to set an error message.

Member Information

Refer to the following for examples of each member of the class:

- EditDefectItem [p. 33]
- GetDefectFieldListFromTool [p. 20]
- GetDefectListFromTool [p. 24]
- LaunchDefectTool [p. 32]

- SubmitDefectToTool [p. 16]
- TestConnection [p. 14]

TestConnection

Tests the connection to the defect tool by using the integration parameters provided in the defect Tool Domain.

Syntax

`TestConnection(string inputXML)`

Input XML

The input string is an XML string that is automatically generated by QADirector based on the tool and tool domain settings:

```
<root>
   <properties category="defect" id="" name="">
      <property name="edittool" type="text" value="JiraDefectIntegration.dll" />
      <property name="submittool" type="text" value="JiraDefectIntegration.dll" />
      <property name="retrievetool" type="text" value="JiraDefectIntegration.dll" />
      <property name="Database Server" type="text" value="dtwlib4m-073" />
      <property name="Port Number" type="text" value="8090" />
      <property name="User Name" type="text" value="admin" />
      <property name="User Password" type="text" value="admin" />
      <property name="Project Key" type="text" value="" />
   </properties>
 </root>
```

Return Value

Returns output in an XML string. Returncode value of 1 means success and 0 means error occurred.

```
<root>
  <results>
    <result name="message" value=""/>
    <result name="returncode" value="1"/>
  </results>
</root>
```

Code Sample

```
public string TestConnection(string inputXML)
{
   int Step = 0;
   string User = "";
   string Password = "";
   string DBServer = "";
   string Port = "";
   string token = "";
   XmlDocument rtnXmlDoc = new XmlDocument();
   string XmlResultContent = "";
   string RS = "";
   Jira.JiraSoapServiceService soapService = new Jira.JiraSoapServiceService();

   try
   {
     //-------------------------------------------------------
     //Retrieve Tool Domain parameters from QADirector
     //-------------------------------------------------------
```

```
    XmlDocument xmlDoc = new XmlDocument();
    xmlDoc.LoadXml(inputXML);
    string path = "/root/properties/property";
    XmlNodeList Nodes = xmlDoc.SelectNodes(path);

    for (int i = 0; i < Nodes.Count; i++)
    {
      switch (Nodes[i].Attributes["name"].Value)
      {
         case "User Name": User = Nodes[i].Attributes["value"].Value; break;
         case "User Password": Password = Nodes[i].Attributes["value"].Value; break;
         case "Server Name": DBServer = Nodes[i].Attributes["value"].Value; break;
         case "Port Number": Port = Nodes[i].Attributes["value"].Value; break;
       }
     }

     Step = 1;

     //---------------------------------------------------
     //Test Login
     //---------------------------------------------------
     soapService.Url = "http://" + DBServer + ":" + Port +
"/rpc/soap/jirasoapservice-v2?wsdl";

     token = "";
     try
     {
       token = soapService.login(User, Password);
     }
     catch (Exception e)
     {
       XmlResultContent = "<result name='message' value='" + CleanMsg(e.Message) + "'
 /><result name='returncode' value='" + ErrorCode + "' />";
       RS = XMLBegin + XMLMiddle + XmlResultContent + XMLEnd;
       return RS;
     }

     RS = XMLBegin + XMLMiddle + XMLEnd;
   }
   catch (Exception ex)
   {
     if (Step == 1)
     {
       XmlResultContent = "<result name='message' value='Could not login to the Jira
instance specified. Check the Jira Product Integration settings. Exception: " +
CleanMsg(ex.Message) + "' /><result name='returncode' value='" + ErrorCode + "' />";
     }
     else
     {
       XmlResultContent = "<result name='message' value='" + CleanMsg(ex.Message) + "'
 /><result name='returncode' value='" + ErrorCode + "' />";
       soapService.logout(token);
     }

     RS = XMLBegin + XMLMiddle + XmlResultContent + XMLEnd;
     }

     finally
     {
       if (token != null)
       {
         try
         {
           soapService.logout(token);
         }
         catch
         {}
       }
     }
   return RS;
```

```
  }
```

SubmitDefectToTool

Submits a defect to the defect tool.

Syntax

`SubmitDefectToTool(string inputXML)`

Parameters

The input string is an XML string that is auto generated by QADirector based on the tool and tool domain settings. The tool domain integration parameters and submit defect field values for the selected fields are passed into the input xml.

```
<root>
    <properties category="defect" id="" name="">
       <property name="toolcomponent" type="text" value="JiraDefectIntegration.dll" />

        <property name="projectid" type="text" value="108" />
        <property name="ssoname" type="text" value="admin" />
        <property name="ssopassword" type="text" value="admin" />
        <property name="Database Server" type="text" value="dtwlib4m-073" />
        <property name="Port Number" type="text" value="8090" />
        <property name="User Name" type="text" value="admin" />
        <property name="User Password" type="text" value="admin" />
        <property name="Project Key" type="text" value="TESTING" />
    </properties>
    <fields category="defect" id="" name="">
       <field category="text" name="PropagatedRequirements" datatype="text" value="New
 Requirement" selected="1" fieldtype="" />
        <field category="text" name="DefectSummary" datatype="text" value="Jira Test"
selected="1" fieldtype="" />
       <field category="text" name="DefectDescription" datatype="text" value="Description
 of the TEST" selected="1" fieldtype="" />
      <field category="text" name="JobName" datatype="text" value="Jira Test Job Name"
 selected="1" fieldtype="" />
       <field category="text" name="JobId" datatype="text" value="5" selected="1"
fieldtype="" />
       <field category="text" name="AssetId" datatype="text" value="117" selected="1"
 fieldtype="" />
       <field category="text" name="AssetName" datatype="text" value="Jira Test"
selected="1" fieldtype="" />
      <field category="text" name="AssetType" datatype="text" value="Test" selected="1"
 fieldtype="" />
      <field category="text" name="AssetDescription" datatype="text" value="Description
 of the TEST" selected="1" fieldtype="" />
      <field category="text" name="InstanceId" datatype="text" value="16" selected="1"
 fieldtype="" />
       <field category="text" name="FailureDescription" datatype="text" value="this is
 a failure description" selected="1" fieldtype="" />
       <field category="text" name="TestingToolName" datatype="text" value=""
selected="1" fieldtype="" />
      <field category="text" name="ToolDomainName" datatype="text" value="" selected="1"
 fieldtype="" />
      <field category="text" name="TestExecutedBy" datatype="text" value="" selected="1"
 fieldtype="" />
      <field category="text" name="TestStartTime" datatype="text" value="" selected="1"
 fieldtype="" />
       <field category="text" name="TestEndTime" datatype="text" value="" selected="1"
 fieldtype="" />
       <field category="text" name="ExecutionPlanName" datatype="text"
value="EP.PlanName..Jira" selected="1" fieldtype="" />
       <field category="text" name="ExecutionMachineName" datatype="text" value=""
selected="1" fieldtype="" />
       <field category="text" name="ExecutionMachineOS" datatype="text" value=""
```

```
selected="1" fieldtype="" />
    </fields>
</root>
```

Return Value

Returns output in an XML string. Returncode value of 1 means success and 0 means error occurred.

```
<root>
    <results>
       <result name="message" value="" />
       <result name="returncode" value="1" />
       <result name="defectid" value="10052" />
       <result name="displayid" value="TESTING-45" />
    </results>
</root>
```

Code Sample

```
public string SubmitDefectToTool(string inputXML)
{
  #region variable declarations
  int Step = 0;
  string RS = "";
  string User = "";
  string Password = "";
            string DBServer = "";
  string Port = "";
  string ProjectKey = "";
  string token = "";
  string SSOUser = "";
  string SSOPassword = "";
  string XmlContent = "";
  string strCommon = "-----QAD Specific Fields-----\n";
  string strCommonExec = "---Execution Details---\n";
  string strDescription = "";
  string strPropogatedTR = "";
  string strTestDesc = "";
  string strTestOwnerName = "";
  string strJobName = "";
  string strScriptName = "";
  string strScriptDesc = "";
  string strFailureDesc = "";
  string strStatus = "";
  string strDefectSummary = "";
  string strStartTime = "";
  string strEndTime = "";
  string strExecMach = "";
#endregion

  Jira.JiraSoapServiceService soapService = new Jira.JiraSoapServiceService();
  Jira.RemoteIssue issue = new JiraDefectIntegration.Jira.RemoteIssue();
  Jira.RemoteComment comment = new JiraDefectIntegration.Jira.RemoteComment();

  try
  {
      XmlDocument xmlDoc = new XmlDocument();
      xmlDoc.LoadXml(inputXML);
      string path = "/root/properties/property";
      XmlNodeList Nodes = xmlDoc.SelectNodes(path);

      //-------------------------------------------------------
      //Retrieve Tool Domain parameters from QADirector
      //-------------------------------------------------------
      for (int i = 0; i < Nodes.Count; i++)
      {

      switch (Nodes[i].Attributes["name"].Value)
      {
       case "User Name":
           User = Nodes[i].Attributes["value"].Value; break;
```

```
      case "User Password":
          Password = Nodes[i].Attributes["value"].Value; break;
      case "Server Name":
          DBServer = Nodes[i].Attributes["value"].Value; break;
      case "Port Number":
          Port = Nodes[i].Attributes["value"].Value; break;
      case "Project Key":
          ProjectKey = Nodes[i].Attributes["value"].Value; break;
      case "ssoname":
          SSOUser = Nodes[i].Attributes["value"].Value; break;
      case "ssopassword":
          SSOPassword = Nodes[i].Attributes["value"].Value; break;
      }
     }

     path = "/root/fields/field";
     Nodes = xmlDoc.SelectNodes(path);

     //--------------------------------------------------------
     //Retrieve information to Submit Issue
     //--------------------------------------------------------
     for (int i = 0; i < Nodes.Count; i++)
     {
  if (Nodes[i].Attributes["category"].Value != "file")
  {
   switch (EscapeQuotes(Nodes[i].Attributes["name"].Value))
   {
       case SubmitDefectFields.DefectSynopsis:
           strDefectSummary = EscapeQuotes(Nodes[i].Attributes["value"].Value); break;

       case SubmitDefectFields.Status:
            strStatus = EscapeQuotes(Nodes[i].Attributes["value"].Value);
            issue.status = strStatus;  break;
       case SubmitDefectFields.DefectDescription:
           strDescription = EscapeQuotes(Nodes[i].Attributes["value"].Value) + "\n\n";
break;
       case SubmitDefectFields.PropagatedTRs:
            strPropogatedTR += EscapeQuotes(Nodes[i].Attributes["value"].Value);
            if (strPropogatedTR == string.Empty)
            { strPropogatedTR = "  -  "; }
            strCommon += "Test Requirments: " + strPropogatedTR + "\n";  break;
       case SubmitDefectFields.TestDesc:
            strTestDesc = EscapeQuotes(Nodes[i].Attributes["value"].Value);
            if (strTestDesc == string.Empty)
            { strTestDesc = "  -  "; }
            strCommon += "Test Summary: " + strTestDesc + "\n"; break;
       case SubmitDefectFields.TestOwnerName:
            strTestOwnerName = EscapeQuotes(Nodes[i].Attributes["value"].Value);
            strCommon += "Test Owner Name: " + strTestOwnerName + "\n"; break;
       case SubmitDefectFields.JobName:
            strJobName = EscapeQuotes(Nodes[i].Attributes["value"].Value);
            if (strJobName == string.Empty)
            { strJobName = "  -  "; }
            strCommon += "Job Name: " + strJobName + "\n";
       case SubmitDefectFields.ScriptName:
            strScriptName = EscapeQuotes(Nodes[i].Attributes["value"].Value);
            if (strScriptName == string.Empty)
            { strScriptName = "  -  "; }
            strCommon += "Script Name: " + strScriptName + "\n"; break;
       case SubmitDefectFields.ScriptDesc:
            strScriptDesc = EscapeQuotes(Nodes[i].Attributes["value"].Value);
            if (strScriptDesc == string.Empty)
            { strScriptDesc = "  -  "; }
            strCommon += "Script Description: " + strScriptDesc + "\n"; break;
       case SubmitDefectFields.FailureDesc:
            strFailureDesc = EscapeQuotes(Nodes[i].Attributes["value"].Value);
            if (strFailureDesc == string.Empty)
            { strFailureDesc = "  -  "; }
            strCommon += "Failure Description: " + strFailureDesc + "\n"; break;
        case SubmitDefectFields.ExecMachine:
            strExecMach = EscapeQuotes(Nodes[i].Attributes["value"].Value);
            if (strExecMach == string.Empty)
            { strExecMach = "  -  "; }
            issue.environment = "Execution Machine: " + strExecMach + "\n";
```

```
             strCommonExec += "Execution Machine Name: " + strExecMach + "\n";  break;
        case SubmitDefectFields.StartTime:
             strStartTime = EscapeQuotes(Nodes[i].Attributes["value"].Value);
             strCommonExec += "Test Start: " + strStartTime + "\n"; break;
        case SubmitDefectFields.EndTime:
             strEndTime = EscapeQuotes(Nodes[i].Attributes["value"].Value.ToString());
             strCommonExec += "Test End: " + strEndTime + "\n";  break;
      }
     }
   }

      issue.type = "1"; // Makes the issue type a "Bug"
      issue.description = strDescription + strCommon + strCommonExec;
      issue.summary = strDefectSummary;

      Step = 1;

      //-----------------------------------------------------
      //Login
      //-----------------------------------------------------
      soapService.Url = "http://" + DBServer + ":" + Port +
"/rpc/soap/jirasoapservice-v2?wsdl";
      token = "";
      try
      {
         token = soapService.login(SSOUser, SSOPassword);
      }
      catch (Exception e)
      {
         XmlContent = "<result name='message' value='" + CleanMsg(e.Message) + "'
/><result name='returncode' value='" + ErrorCode + "' />";
         RS = XMLBegin + XMLMiddle + XmlContent + XMLEnd;
         return RS;
      }

      Step = 2;

      //-----------------------------------------------------
      //Retrieve Projects & Submit Issue
      //-----------------------------------------------------
      try
      {
        issue.project = ProjectKey;

        //find the project
        Jira.RemoteProject project = new JiraDefectIntegration.Jira.RemoteProject();
        project = soapService.getProjectByKey(token, ProjectKey);
        if (project != null)
        {   //if the project is found, assign this issue to the team lead (which is
default in JIRA)
           issue.assignee = project.lead;
        }
        else
        {   //if the project is not found, assign this issue to the current
            // user so that the issue is created with out any problem
            issue.assignee = SSOUser;
         }

      Jira.RemoteIssue createdIssue;
      createdIssue = soapService.createIssue(token, issue);
     XmlContent = "<result name='message' value='' /><result name='returncode' value='"
 + SuccessCode + "' /><result name='defectid' value='" + createdIssue.id + "' /><result
 name='displayid' value='" + createdIssue.key + "' />";
      }
      catch (Exception ex)
      {
          XmlContent = "<result name='message' value='Defect submission failure. " +
CleanMsg(ex.Message) + "' /><result name='returncode' value='" + ErrorCode + "' /><result
 name='defectid' value='" + RS + "' />";
      }
     }
     catch (Exception ex)
     {
      if (Step == 1)
       XmlContent = "<result name='message' value='Could not login to the Jira instance
```

```
 specified. Check the Product Integration settings. Exception:" + CleanMsg(ex.Message)
 + "' /><result name='returncode' value='" + ErrorCode + "' /><result name='defectid'
value='" + RS + "' />";
      else
      {
     if (Step == 2)
     {
        XmlContent = "<result name='message' value='Unable to submit item.  Required
Fields not set correctly for Tool Domain. Exception:" + CleanMsg(ex.Message) + "'
/><result name='returncode' value='" + ErrorCode + "' /><result name='defectid' value='"
 + RS + "' />";
     }
     else
        XmlContent = "<result name='message' value='" + CleanMsg(ex.Message) + "'
/><result name='returncode' value='" + ErrorCode + "' /><result name='defectid' value='"
 + RS + "' /><result name='displayid' value='" + RS + "' />";
      }
     }
    finally
    {
      soapService.logout(token);
     }

     RS = XMLBegin + XMLMiddle + XmlContent + XMLEnd;
     return RS;
 }
```

GetDefectFieldListFromTool

Retrieves the fields information of the defect item in the defect tool.

Syntax

`GetDefectFieldListFromTool(string inputXML)`

Parameters

`inputXML` is an xml string.

```
<root>
    <properties category="defect" id="" name="">
    <property name="toolcomponent" type="text" value="JiraDefectIntegration.dll" />
    <property name="Database Server" type="text" value="dtwlib4m-073" />
    <property name="Port Number" type="text" value="8090" />
    <property name="User Name" type="text" value="admin" />
    <property name="User Password" type="text" value="admin" />
    <property name="Project Key" type="text" value="" />
    </properties>
</root>
```

Return Value

Returns a list of defect fields in an XML string. Returncode value of 1 means success and 0 means an error occurred.

```
<root>
    <fields>
        <field name="key" datatype="TEXT" selected="1" reqfieldtype="DISPLAYID" />
        <field name="id" datatype="NUMERIC" selected="1" reqfieldtype="UNIQUEID" />
        <field name="Type" datatype="TEXT" selected="0" reqfieldtype="OTHER"
caption="Type" />
        <field name="status" datatype="TEXT" selected="1" reqfieldtype="STATUS" />
        <field name="resolution" datatype="NUMERIC" selected="0" reqfieldtype="OTHER"
caption="Resolution" />
        <field name="priority" datatype="TEXT" selected="1" reqfieldtype="PRIORITY" />

        <field name="Assignee" datatype="TEXT" selected="0" reqfieldtype="OTHER"
```

```
caption="Assignee" />
        <field name="Reporter" datatype="TEXT" selected="0" reqfieldtype="OTHER"
caption="Reporter" />
        <field name="summary" datatype="TEXT" selected="1" reqfieldtype="SUMMARY" />
        <field name="Enviroment" datatype="TEXT" selected="0" reqfieldtype="OTHER"
caption="Enviroment" />
        <field name="Description" datatype="TEXT" selected="0" reqfieldtype="OTHER"
caption="Description" />
        <field name="Comments" datatype="TEXT" selected="0" reqfieldtype="OTHER"
caption="Comments" />
        <field name="DueDate" datatype="TEXT" selected="0" reqfieldtype="OTHER"
caption="DueDate" />
       <field name="datecreated" datatype="DATE" selected="1" reqfieldtype="DATECREATED"
 />
    </fields>
</root>
```

Example

```
public string GetDefectFieldListFromTool(string inputXML)
        {
            string SelectedYes = "1";
            string SelectedNo = "0";
            int step = 0;
            string User = "";
            string Password = "";
            string DBServer = "";
            string Port = "";
            string ProjectKey = "";
            string token = "";
            string SSOUser = "";
            string SSOPassword = "";
            string XmlResultContent = "";
            XmlDocument rtnXmlDoc = new XmlDocument();
            string RS = "";

           Jira.JiraSoapServiceService soapService = new Jira.JiraSoapServiceService();


            try
            {
                XmlDocument xmlDoc = new XmlDocument();
                xmlDoc.LoadXml(inputXML);
                string path = "/root/properties/property";
                XmlNodeList Nodes = xmlDoc.SelectNodes(path);

                //---------------------------------------------------------
                //Retrieve Tool Domain parameters from QADirector
                //---------------------------------------------------------
                for (int i = 0; i < Nodes.Count; i++)
                {
                    switch (Nodes[i].Attributes["name"].Value)
                    {

                        case "User Name":
                            User = Nodes[i].Attributes["value"].Value;
                            break;
                        case "User Password":
                            Password = Nodes[i].Attributes["value"].Value;
                            break;
                        case "Server Name":
                            DBServer = Nodes[i].Attributes["value"].Value;
                            break;
                        case "Port Number":
                            Port = Nodes[i].Attributes["value"].Value;
                            break;
                        case "Project Key":
                            ProjectKey = Nodes[i].Attributes["value"].Value;
                            break;
                        case "ssoname":
                            SSOUser = Nodes[i].Attributes["value"].Value;
                            break;
                        case "ssopassword":
                            SSOPassword = Nodes[i].Attributes["value"].Value;
```

```
                            break;
                    }
                }

                XmlElement RootElem = rtnXmlDoc.CreateElement("root");
                rtnXmlDoc.AppendChild(RootElem);

                step = 1;

                //---------------------------------------------------
                //Test Login
                //---------------------------------------------------
                soapService.Url = "http://" + DBServer + ":" + Port +
"/rpc/soap/jirasoapservice-v2?wsdl";
                token = "";
                try
                {
                    token = soapService.login(User, Password);
                }
                catch (Exception e)
                {
                    XmlResultContent = "<result name='message' value='" +
CleanMsg(e.Message) + "' /><result name='returncode' value='" + ErrorCode + "' />";
                    RS = XMLBegin + XMLMiddle + XmlResultContent + XMLEnd;
                    return RS;
                }

                step = 2;

                //---------------------------------------------------
                //Gather Tool Domain Data
                //---------------------------------------------------
                XmlElement FieldsElem = rtnXmlDoc.CreateElement("fields");
                RootElem.AppendChild(FieldsElem);

                StringCollection tmpFields = new StringCollection();
                tmpFields.AddRange(getFieldNames());

                for (int j = 0; j < tmpFields.Count; j++)
                {
                    string field = tmpFields[j];

                    XmlElement FieldElem = this.CreateFieldNode(rtnXmlDoc,
EncodeXmlValue(field), "", SelectedNo, "");

                    if (field.CompareTo(JiraFields.KEY) == 0)
                    {
                        FieldElem.SetAttribute(XMLConstants.FIELD_DATATYPE_ATTR,
FieldDataTypes.Text);
                        FieldElem.SetAttribute(XMLConstants.FIELD_REQFIELDTYPE_ATTR,
RequiredFieldTypes.DisplayID);
                        FieldElem.SetAttribute(XMLConstants.FIELD_SELECTED_ATTR,
SelectedYes);
                    }
                    else if (field.CompareTo(JiraFields.ID) == 0)
                    {
                        FieldElem.SetAttribute(XMLConstants.FIELD_DATATYPE_ATTR,
FieldDataTypes.Numeric);
                        FieldElem.SetAttribute(XMLConstants.FIELD_REQFIELDTYPE_ATTR,
RequiredFieldTypes.UniqueID);
                        FieldElem.SetAttribute(XMLConstants.FIELD_SELECTED_ATTR,
SelectedYes);
                    }
                    else if (field.CompareTo(JiraFields.SUMMARY) == 0)
                    {
                        FieldElem.SetAttribute(XMLConstants.FIELD_DATATYPE_ATTR,
FieldDataTypes.Text);
                        FieldElem.SetAttribute(XMLConstants.FIELD_REQFIELDTYPE_ATTR,
RequiredFieldTypes.Summary);
                        FieldElem.SetAttribute(XMLConstants.FIELD_SELECTED_ATTR,
SelectedYes);
                    }
                    else if (field.CompareTo(JiraFields.STATUS) == 0)
                    {
                        FieldElem.SetAttribute(XMLConstants.FIELD_DATATYPE_ATTR,
```

```
FieldDataTypes.Text);
                        FieldElem.SetAttribute(XMLConstants.FIELD_REQFIELDTYPE_ATTR,
RequiredFieldTypes.Status);
                        FieldElem.SetAttribute(XMLConstants.FIELD_SELECTED_ATTR,
SelectedYes);
                    }
                    else if (field.CompareTo(JiraFields.PRIORITY) == 0)
                    {
                        FieldElem.SetAttribute(XMLConstants.FIELD_DATATYPE_ATTR,
FieldDataTypes.Text);
                        FieldElem.SetAttribute(XMLConstants.FIELD_REQFIELDTYPE_ATTR,
RequiredFieldTypes.Priority);
                        FieldElem.SetAttribute(XMLConstants.FIELD_SELECTED_ATTR,
SelectedYes);
                    }
                    else if (field.CompareTo(JiraFields.RESOLUTION) == 0)
                    {
                        FieldElem.SetAttribute(XMLConstants.FIELD_DATATYPE_ATTR,
FieldDataTypes.Numeric);
                        FieldElem.SetAttribute(XMLConstants.FIELD_REQFIELDTYPE_ATTR,
RequiredFieldTypes.Other);
                        FieldElem.SetAttribute(XMLConstants.FIELD_NAME_ATTR,
"resolution");
                        FieldElem.SetAttribute(XMLConstants.FIELD_CAPTION_ATTR,
"Resolution");
                        FieldElem.SetAttribute(XMLConstants.FIELD_SELECTED_ATTR,
SelectedNo);
                    }
                    else if (field.CompareTo(JiraFields.DATE_CREATED) == 0)
                    {
                        FieldElem.SetAttribute(XMLConstants.FIELD_DATATYPE_ATTR,
FieldDataTypes.Date);
                        FieldElem.SetAttribute(XMLConstants.FIELD_REQFIELDTYPE_ATTR,
RequiredFieldTypes.DateCreated);
                        FieldElem.SetAttribute(XMLConstants.FIELD_SELECTED_ATTR,
SelectedYes);
                    }
                    else if (field.CompareTo(JiraFields.DUE_DATE) == 0)
                    {
                        FieldElem.SetAttribute(XMLConstants.FIELD_DATATYPE_ATTR,
FieldDataTypes.Text);
                        FieldElem.SetAttribute(XMLConstants.FIELD_REQFIELDTYPE_ATTR,
RequiredFieldTypes.Other);
                       FieldElem.SetAttribute(XMLConstants.FIELD_NAME_ATTR, "DueDate");

                        FieldElem.SetAttribute(XMLConstants.FIELD_CAPTION_ATTR,
"DueDate");
                        FieldElem.SetAttribute(XMLConstants.FIELD_SELECTED_ATTR,
SelectedNo);
                    }
                    else
                    {
                        FieldElem.SetAttribute(XMLConstants.FIELD_DATATYPE_ATTR,
FieldDataTypes.Text);
                        FieldElem.SetAttribute(XMLConstants.FIELD_NAME_ATTR, field);
                       FieldElem.SetAttribute(XMLConstants.FIELD_CAPTION_ATTR, field);

                        FieldElem.SetAttribute(XMLConstants.FIELD_REQFIELDTYPE_ATTR,
RequiredFieldTypes.Other);
                    }
                    FieldsElem.AppendChild(FieldElem);
                }

                if (FieldsElem == null)
                {
                    XmlResultContent = "<result name='message' value='Could not find
the specified Jira Product in the  database.' /><result name='returncode' value='" +
ErrorCode + "' />";
                    RS = XMLBegin + XMLMiddle + XmlResultContent + XMLEnd;

                    if (soapService.login(User, Password) == null)
                    {
                        soapService.logout(token);
                    }
```

```
                    return RS;
                }

                RS = rtnXmlDoc.InnerXml;

            }
            catch (Exception ex)
            {
                if (step == 1)
                {
                    XmlResultContent = "<result name='message' value='Could not login
to the Jira instance specified. Check the Product Integration settings. Exception: " +
 CleanMsg(ex.Message) + "' /><result name='returncode' value='" + ErrorCode + "' />";
                }
                else
                {
                    XmlResultContent = "<result name='message' value='" +
CleanMsg(ex.Message) + "' /><result name='returncode' value='" + ErrorCode + "' />";
                }

                RS = XMLBegin + XMLMiddle + XmlResultContent + XMLEnd;
            }
            finally
            {
                try
                {
                    soapService.logout(token);
                }
                catch
                {}

            }

            return RS;
        }
```

GetDefectListFromTool

Retrieves the defect data from the defect tool.

Syntax

`GetDefectListFromTool(string inputXML)`

Parameters

`string inputXML`

```
<root>
    <fields>
        <field name="key" datatype="TEXT" selected="1" reqfieldtype="DISPLAYID" />
        <field name="id" datatype="NUMERIC" selected="1" reqfieldtype="UNIQUEID" />
        <field name="Type" datatype="TEXT" selected="0" reqfieldtype="OTHER"
caption="Type" />
        <field name="status" datatype="TEXT" selected="1" reqfieldtype="STATUS" />
        <field name="resolution" datatype="NUMERIC" selected="0" reqfieldtype="OTHER"
caption="Resolution" />
        <field name="priority" datatype="TEXT" selected="1" reqfieldtype="PRIORITY" />

        <field name="Assignee" datatype="TEXT" selected="0" reqfieldtype="OTHER"
caption="Assignee" />
        <field name="Reporter" datatype="TEXT" selected="0" reqfieldtype="OTHER"
caption="Reporter" />
        <field name="summary" datatype="TEXT" selected="1" reqfieldtype="SUMMARY" />
        <field name="Enviroment" datatype="TEXT" selected="0" reqfieldtype="OTHER"
caption="Enviroment" />
        <field name="Description" datatype="TEXT" selected="0" reqfieldtype="OTHER"
caption="Description" />
        <field name="Comments" datatype="TEXT" selected="0" reqfieldtype="OTHER"
```

```
caption="Comments" />
        <field name="DueDate" datatype="TEXT" selected="0" reqfieldtype="OTHER"
caption="DueDate" />
       <field name="datecreated" datatype="DATE" selected="1" reqfieldtype="DATECREATED"
 />
      </fields>
</root>
```

Return Value

Returns an XML string with defects and field information. Returncode value of 1 means success and 0 means error occurred. The following is an example, see **XML Schema Return Format** below for the full schema.

```
<root>
   <items>
      <item id="10051">
         <fields>
            <field name="key" value="TESTING-44" />
            <field name="id" value="10051" />
            <field name="status" value="Open" />
            <field name="priority" value="" />
            <field name="summary" value="Jira Test" />
            <field name="datecreated" value="7/29/2008" />
         </fields>
      </item>
      <item id="10050">
         <fields>
            <field name="key" value="TESTING-43" />
            <field name="id" value="10050" />
            <field name="status" value="Open" />
            <field name="priority" value="" />
            <field name="summary" value="Jira Test" />
            <field name="datecreated" value="7/29/2008" />
         </fields>
      </item>
      <item id="10049">
         <fields>
            <field name="key" value="TESTING-42" />
            <field name="id" value="10049" />
            <field name="status" value="Open" />
            <field name="priority" value="" />
            <field name="summary" value="Jira Test" />
            <field name="datecreated" value="7/29/2008" />
         </fields>
      </item>
   </items>
</root>
```

XML Schema Return Format

The output XML is required to adhere to the following schema:

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="root">
      <xs:complexType>
         <xs:sequence>
            <!-- items section begin -->
            <xs:element name="items" minOccurs="0" maxOccurs="1">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                           <xs:sequence>
                              <xs:element name="fields" minOccurs="0" maxOccurs="1">
                                 <xs:complexType>
                                    <xs:sequence>
                                       <xs:element name="field" minOccurs="0"
maxOccurs="unbounded">
                                          <xs:complexType>
```

```
                                                <xs:attribute name="category"
type="xs:string" use="optional"></xs:attribute>
                                                <xs:attribute name="name" type="xs:string"
 use="required"></xs:attribute>
                                             <xs:attribute name="datatype" use="optional">

                                                   <xs:simpleType>
                                                      <xs:restriction base="xs:string">
                                                         <xs:enumeration
value="TEXT"></xs:enumeration>
                                                         <xs:enumeration
value="NUMERIC"></xs:enumeration>
                                                         <xs:enumeration
value="DATE"></xs:enumeration>
                                                      </xs:restriction>
                                                   </xs:simpleType>
                                                </xs:attribute>
                                               <xs:attribute name="value" type="xs:string"
 use="optional"></xs:attribute>
                                             <xs:attribute name="selected" use="optional">

                                                   <xs:simpleType>
                                                      <xs:restriction base="xs:string">
                                                         <xs:enumeration
value="0"></xs:enumeration>
                                                         <xs:enumeration
value="1"></xs:enumeration>
                                                      </xs:restriction>
                                                   </xs:simpleType>
                                                </xs:attribute>
                                                <xs:attribute name="reqfieldtype"
use="optional">
                                                   <xs:simpleType>
                                                      <xs:restriction base="xs:string">
                                                         <xs:enumeration
value="UNIQUEID"></xs:enumeration>
                                                         <xs:enumeration
value="DISPLAYID"></xs:enumeration>
                                                         <xs:enumeration
value="STATUS"></xs:enumeration>
                                                         <xs:enumeration
value="PRIORITY"></xs:enumeration>
                                                         <xs:enumeration
value="SUMMARY"></xs:enumeration>
                                                         <xs:enumeration
value="DATECREATED"></xs:enumeration>
                                                         <xs:enumeration
value="OTHER"></xs:enumeration>
                                                      </xs:restriction>
                                                   </xs:simpleType>
                                                </xs:attribute>
                                             </xs:complexType>
                                          </xs:element>
                                       </xs:sequence>
                                    </xs:complexType>
                                 </xs:element>
                              </xs:sequence>
                              <xs:attribute name="id" type="xs:string"
use="required"></xs:attribute>
                           </xs:complexType>
                        </xs:element>
                     </xs:sequence>
                  </xs:complexType>
               </xs:element>
               <!-- items section complete -->
               <!-- Results section begin -->
               <xs:element name="results" minOccurs="0" maxOccurs="1">
                  <xs:complexType>
                     <xs:sequence>
                        <xs:element name="result" minOccurs="0" maxOccurs="unbounded">
                           <xs:complexType>
                              <xs:attribute name="name" use="required">
                                 <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                       <xs:enumeration value="message"></xs:enumeration>
```

```
                                <xs:enumeration value="returncode"></xs:enumeration>

                                  <xs:enumeration value="defectid"></xs:enumeration>

                                </xs:restriction>
                              </xs:simpleType>
                            </xs:attribute>
                            <xs:attribute name="value" type="xs:string"
use="required"></xs:attribute>
                          </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <!-- Results section complete-->
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Code Sample

```
public string GetDefectListFromTool(string inputXML)
        {
            string RS = inputXML;
            int Step = 0;
            string User = "";
            string Password = "";
            string DBServer = "";
            string Port = "";
            string ProjectKey = "";
            string token = "";
            string XmlResultContent = "";
            XmlDocument rtnXmlDoc = new XmlDocument();
           Jira.JiraSoapServiceService soapService = new Jira.JiraSoapServiceService();

            bool bProjectFound = false;

            try
            {
                XmlDocument xmlDoc = new XmlDocument();
                xmlDoc.LoadXml(inputXML);
                string path = "/root/properties/property";
                XmlNodeList Nodes = xmlDoc.SelectNodes(path);

                //---------------------------------------------------------
                //Retrieve Tool Domain parametets from QADirector
                //---------------------------------------------------------
                for (int i = 0; i < Nodes.Count; i++)
                {
                    switch (Nodes[i].Attributes["name"].Value)
                    {
                        case "User Name":
                            User = Nodes[i].Attributes["value"].Value;
                            break;
                        case "User Password":
                            Password = Nodes[i].Attributes["value"].Value;
                            break;
                        case "Server Name":
                            DBServer = Nodes[i].Attributes["value"].Value;
                            break;
                        case "Port Number":
                            Port = Nodes[i].Attributes["value"].Value;
                            break;
                        case "Project Key":
                            ProjectKey = Nodes[i].Attributes["value"].Value;
                            break;
                    }
                }

                XmlElement RootElem = rtnXmlDoc.CreateElement("root");
                rtnXmlDoc.AppendChild(RootElem);

                Step = 1;
```

```
                //------------------------------------------------------
                //Login
                //------------------------------------------------------
                soapService.Url = "http://" + DBServer + ":" + Port +
"/rpc/soap/jirasoapservice-v2?wsdl"; ;

                token = "";
                try
                {
                    token = soapService.login(User, Password);
                }
                catch (Exception e)
                {
                    XmlResultContent = "<result name='message' value='" +
CleanMsg(e.Message) + "' /><result name='returncode' value='" + ErrorCode + "' />";
                    RS = XMLBegin + XMLMiddle + XmlResultContent + XMLEnd;
                    return RS;
                }

                Step = 2;

               Jira.RemoteIssue[] issues = new JiraDefectIntegration.Jira.RemoteIssue[]
 { };
                Jira.RemoteProject project = new
JiraDefectIntegration.Jira.RemoteProject();

                XmlElement ItemsElem = rtnXmlDoc.CreateElement("items");
                RootElem.AppendChild(ItemsElem);

                project = soapService.getProjectByKey(token, ProjectKey);

                if (project != null)
                {
                    bProjectFound = true;
                }

                if (bProjectFound)
                {
                    //get all of the issues for this project
                    try
                    {
                        issues = soapService.getIssuesFromTextSearchWithProject(token,
 new string[] { project.key }, "", 999999999);
                        foreach (Jira.RemoteIssue issue in issues)
                        {
                            // Spit out the XML

                            // First, the defect node
                            XmlElement ItemElem = this.CreateItemNode(rtnXmlDoc,
issue.id.ToString());
                            ItemsElem.AppendChild(ItemElem);

                            // Now, the field nodes
                            XmlElement FieldsElem = rtnXmlDoc.CreateElement("fields");
                            ItemElem.AppendChild(FieldsElem);

                            // Loop thru all possible fields of a bug
                            StringCollection fldNames = new StringCollection();
                            fldNames.AddRange(getFieldNames());
                            for (int j = 0; j < fldNames.Count; j++)
                            {
                                //Find this field name in the fields xml
                                XmlNode FieldNode =
xmlDoc.SelectSingleNode(XMLConstants.ROOT_NODE + "/" +
                                                        XMLConstants.FIELDS_NODE + "/" +
XMLConstants.FIELD_NODE +
                                                       "[@" + XMLConstants.FIELD_NAME_ATTR + "='"
 + fldNames[j] + "']");

                                if (FieldNode != null)
                                {
                                    if (fldNames[j].CompareTo("id") == 0)
                                    {
```

```
FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue(issue.id)));
                                    }
                                    else if (fldNames[j].CompareTo("summary") == 0)
                                    {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue(issue.summary)));
                                    }
                                    else if (fldNames[j].CompareTo("key") == 0)
                                    {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue(issue.key)));
                                    }
                                    else if (fldNames[j].CompareTo("Type") == 0)
                                    {
                                        #region Type Names
                                        if (issue.type.CompareTo("1") == 0)
                                        {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue("Bug")));
                                        }
                                        else if (issue.type.CompareTo("2") == 0)
                                        {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue("New Feature")));
                                        }
                                        else if (issue.type.CompareTo("3") == 0)
                                        {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue("Task")));
                                        }
                                        else if (issue.type.CompareTo("4") == 0)
                                        {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue("Improvment")));
                                        }
                                        #endregion
                                    }
                                   else if (fldNames[j].CompareTo("datecreated") == 0)

                                    {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue(issue.created.Value.ToShortDateString())));
                                    }
                                    else if (fldNames[j].CompareTo("status") == 0)
                                    {
                                        #region Status Names
                                        if (issue.status.Equals("1"))
                                        {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue("Open")));
                                        }
                                        else if (issue.status.Equals("3"))
                                        {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j], EncodeXmlValue("In
 Progress")));
                                        }
                                        else if (issue.status.Equals("4"))
                                        {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue("Reopened")));
                                        }
                                        else if (issue.status.Equals("5"))
                                        {
```

```
FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue("Resolved")));
                                    }
                                    else if (issue.status.Equals("6"))
                                    {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue("Closed")));
                                    }
                                    #endregion
                                }
                                else if (fldNames[j].CompareTo("Resolution") == 0)
                                {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue(issue.resolution)));
                                }
                                else if (fldNames[j].CompareTo("priority") == 0)
                                {
                                    #region Priority Names
                                    try
                                    {
                                        if (issue.priority.EndsWith("1"))
                                        {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue("Blocker")));
                                        }
                                        else if (issue.priority.EndsWith("2"))
                                        {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue("Critical")));
                                        }
                                        else if (issue.priority.EndsWith("3"))
                                        {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue("Major")));
                                        }
                                        else if (issue.priority.EndsWith("4"))
                                        {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue("Minor")));
                                        }
                                        else if (issue.priority.EndsWith("5"))
                                        {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue("Trivial")));
                                        }
                                    }
                                    catch
                                    {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j], EncodeXmlValue("
 ")));
                                    }
                                    #endregion
                                }
                                else if (fldNames[j].CompareTo("Assignee") == 0)
                                {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue(issue.assignee)));
                                }
                                else if (fldNames[j].CompareTo("Reporter") == 0)
                                {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue(issue.reporter)));
                                }
                                else if (fldNames[j].CompareTo("Enviroment") == 0)
                                {
```

```
FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue(issue.environment)));
                                    }
                                   else if (fldNames[j].CompareTo("Description") == 0)

                                    {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue(issue.description)));
                                    }
                                    else if (fldNames[j].CompareTo("DueDate") == 0)
                                    {
                                        #region DueDate Specs
                                        if (issue.duedate.Equals(null))
                                        {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue("-")));
                                        }
                                        else
                                        {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue(issue.duedate.Value.ToShortDateString())));
                                        }
                                        #endregion
                                    }
                                    else if (fldNames[j].CompareTo("Comments") == 0)
                                    {
                                        Jira.RemoteComment[] commentarray =
soapService.getComments(token, issue.key);

                                        foreach (Jira.RemoteComment comment in
commentarray)
                                        {

FieldsElem.AppendChild(this.CreateItemFieldNode(rtnXmlDoc, fldNames[j],
EncodeXmlValue(comment.body + " || By: " + comment.author)));
                                        }
                                    }
                                }
                            }
                        }
                    }
                    catch
                    {
                       XmlResultContent = "<result name='message' value='Could not find
 any issues in the specified Jira product " + CleanMsg(ProjectKey) + "in the Jira
database.' /><result name='returncode' value='" + ErrorCode + "' />";
                        RS = XMLBegin + XMLMiddle + XmlResultContent + XMLEnd;
                    }
                }
                else
                {
                    XmlResultContent = "<result name='message' value='Could not find
the specified Jira product \"" + CleanMsg(ProjectKey) + "\" in the Jira database.'
/><result name='returncode' value='" + ErrorCode + "' />";
                    RS = XMLBegin + XMLMiddle + XmlResultContent + XMLEnd;

                    try
                    {
                        soapService.logout(token);
                    }
                    catch
                    { }
                    return RS;
                }
                RS = rtnXmlDoc.InnerXml;
            }

            catch (Exception ex)
            {
                if (Step == 1)
                    XmlResultContent = "<result name='message' value='Could not login
```

```
to the Jira instance specified. Check the Product Integration settings. Exception: " +
 CleanMsg(ex.Message) + "' /><result name='returncode' value='" + ErrorCode + "' /><result
 name='defectid' value='' />";
                else
                {
                    XmlResultContent = "<result name='message' value='" +
CleanMsg(ex.Message) + "' /><result name='returncode' value='" + ErrorCode + "' /><result
 name='defectid' value='' />";
                }

                RS = XMLBegin + XMLMiddle + XmlResultContent + XMLEnd;
            }
            finally
            {
                try
                {
                    soapService.logout(token);
                }
                catch
                {}
            }

            return RS;
        }
```

LaunchDefectTool

Launches the defect tool.

Syntax

`LaunchDefectTool(string inputXML)`

Parameters

The input string is an XML string that is auto generated by QADirector based on the tool and tool domain settings.

```
<root>
    <properties category="defect" id="10026" name="">
       <property name="toolcomponent" type="text" value="JiraDefectIntegration.dll" />

        <property name="defectid" type="text" value="TESTING-19" />
        <property name="projectid" type="text" value="108" />
        <property name="ssoname" type="text" value="admin" />
        <property name="ssopassword" type="text" value="admin" />
        <property name="Database Server" type="text" value="dtwlib4m-073" />
        <property name="Port Number" type="text" value="8090" />
        <property name="User Name" type="text" value="admin" />
        <property name="User Password" type="text" value="admin" />
        <property name="Project Key" type="text" value="TESTING" />
    </properties>
</root>
```

Return Value

Returns output in an xml string. Returncode value of 1 means success and 0 means error occurred.

```
<root>
             <results>
                 <result name="message" value="" />
                 <result name="returncode" value="1" />
```

```
            </results>
         </root>
```

Code Sample

```
public string LaunchDefectTool(string inputXML)
{
   string XmlResultContent = "";
   string rtnXmlStr = "";
   string strJiraURL = "";
   string DBServer = "";
   string Port = "";

   XmlDocument xmlDoc = new XmlDocument();
   xmlDoc.LoadXml(inputXML);
   string path = "/root/properties/property";
   XmlNodeList Nodes = xmlDoc.SelectNodes(path);

   for (int i = 0; i < Nodes.Count; i++)
   {
     switch (Nodes[i].Attributes["name"].Value)
     {
       case "Server Name":
         DBServer = Nodes[i].Attributes["value"].Value;
         break;
       case "Port Number":
         Port = Nodes[i].Attributes["value"].Value;
         break;
     }
   }

   try
   {
     strJiraURL = "http://" + DBServer + ":" + Port;

     if (strJiraURL != "http://:")
     {
       LaunchURL(strJiraURL);
     }

     XmlResultContent = "<result name='message' value='' /><result name='returncode'
value='" + SuccessCode + "' />";
   }
   catch (Exception excep)
   {
     XmlResultContent = "<result name='message' value='\n\nError occurred launching
Jira: \n" + excep.Message + "'  /><result name='returncode' value='" + ErrorCode + "'
/>";
   }

   rtnXmlStr = XMLBegin + XMLMiddle + XmlResultContent + XMLEnd;
   return rtnXmlStr;
}
```

EditDefectItem

Opens the given defect in the defect tool.

Syntax

`EditDefectItem(string inputXML)`

Parameters

`inputXML` is an xml string that is auto generated by QADirector based on the Tool and Tool Domain settings and the requested defect identifier.

```
<root>
   <properties category="defect" id="10026" name="">
      <property name="ischangepoint" type="text" value="0" />
      <property name="toolcomponent" type="text" value="JiraDefectIntegration.dll" />
      <property name="defectid" type="text" value="TESTING-19" />
      <property name="projectid" type="text" value="108" />
     <property name="Site Id" type="text" value="4bc4c8dd-27a2-4b4b-b4c5-75c72df7c5b4"
 />
      <property name="ssoname" type="text" value="admin" />
      <property name="ssopassword" type="text" value="admin" />
      <property name="Database Server" type="text" value="dtwlib4m-073" />
      <property name="Port Number" type="text" value="8090" />
      <property name="User Name" type="text" value="admin" />
      <property name="User Password" type="text" value="admin" />
      <property name="Project Key" type="text" value="TESTING" />
   </properties>
</root>
```

Return Value

Returns an XML string. Returncode value of 1 means success and 0 means an error occurred.

```
<root>
   <results>
      <result name="message" value="" />
      <result name="returncode" value="1" />
   </results>
</root>
```

Code Sample

```
public string EditDefectItem(string inputXML)
        {
            string XmlResultContent = "";
            string rtnXmlStr = "";
            string User = "";
            string Password = "";
            string DBServer = "";
            string Port = "";
            string DefectID = "";
            string strJiraEditURL = "";
            string SSOUser = "";
            string SSOPassword = "";
            XmlDocument xmlDoc = new XmlDocument();
            xmlDoc.LoadXml(inputXML);
            string path = "/root/properties/property";
            XmlNodeList Nodes = xmlDoc.SelectNodes(path);

            for (int i = 0; i < Nodes.Count; i++)
            {
                switch (Nodes[i].Attributes["name"].Value)
                {
                    case "User Name":
                        User = Nodes[i].Attributes["value"].Value;
                        break;
                    case "User Password":
                        Password = Nodes[i].Attributes["value"].Value;
                        break;
                    case "Server Name":
                        DBServer = Nodes[i].Attributes["value"].Value;
                        break;
                    case "Port Number":
                        Port = Nodes[i].Attributes["value"].Value;
                        break;
                     case "ssoname":
                        SSOUser = Nodes[i].Attributes["value"].Value;
                        break;
                     case "ssopassword":
                        SSOPassword = Nodes[i].Attributes["value"].Value;
                        break;
                }
            }
```

```
            XmlDocument xmlDocs = new XmlDocument();
            xmlDocs.LoadXml(inputXML);
            string paths = "/root/properties";
            XmlNodeList Node = xmlDocs.SelectNodes(paths);

            for (int i = 0; i < Node.Count; i++)
            {
                switch (Node[i].Attributes["category"].Value)
                {
                    case "defect":
                        DefectID = Node[i].Attributes["id"].Value;
                        break;
                }
            }

            try
            {
               strJiraEditURL = "http://" + DBServer + ":" + Port +
"/secure/EditIssue!default.jspa?id=" + DefectID + "&os_username=" + SSOUser +
"&os_password=" + SSOPassword;

                if (DefectID != null && DBServer != null && Port != null)
                {
                    LaunchURL(strJiraEditURL);
                    XmlResultContent = "<result name='message' value='' /><result
name='returncode' value='" + SuccessCode + "' />";
                }
            }
            catch (Exception excep)
            {
                XmlResultContent = "<result name='message' value='\n\nError occurred
launching Jira: \n" + excep.Message + "'  /><result name='returncode' value='" + ErrorCode
 + "' />";
            }

            rtnXmlStr = XMLBegin + XMLMiddle + XmlResultContent + XMLEnd;
            return rtnXmlStr;
        }
```

Deployment

1. After coding the integration DLL, the output binaries should be copied to the following location on the QADirector web server machine: `\Compuware\QADirector\TMServices\ThirdPartyIntegrations`. Your DLL will be deployed in the following manner:

   Automated Tools

   Your DLL will be downloaded from the server to the client when needed.

   Defect Tools

   - For defect retrieval, the DLL is executed on the server.
   - For defect editing, the DLL is always downloaded and executed on the client on demand.
   - For defect submission, the default behavior is silent submission on the server. However, the **Tool Properties** dialog box provides an option to submit defects on the client. If this option is selected, then the DLL will be download and executed on the client.

2. Create a **Tool** and **Tool Domain** in QADirector to allow QADirector to send and receive the appropriate information.

- When creating a **Tool**, be sure to select the appropriate type from the **Tool Type** list. For example: for defects, select the **`Defect`** type. For automated tools, select the **`Automated`** type.
- The **Tool/Tool Domain** integration parameters are created and configured based on the needs of the tool/integration:

  Automated Tools
  Parameters can be set to apply at the tool domain or the tool properties. By applying a parameter at the tool properties, a value can be set that can be used in the tool. Additional Parameters can be added.

  Defect Tools
  Parameters for defect tool domains can be set to apply at the tool domain or the project level. By applying a parameter at the project level, duplicate tool domains can be avoided. The integration parameter values are set in two locations. Parameters that apply across the tool domain are set in the tool domain properties integration parameters tab, while parameters that apply to projects are set in the project properties defect tracking tab.

- **Single Sign On**

  Automated Tools
  You can optionally use **Single Sign On** for the integration.

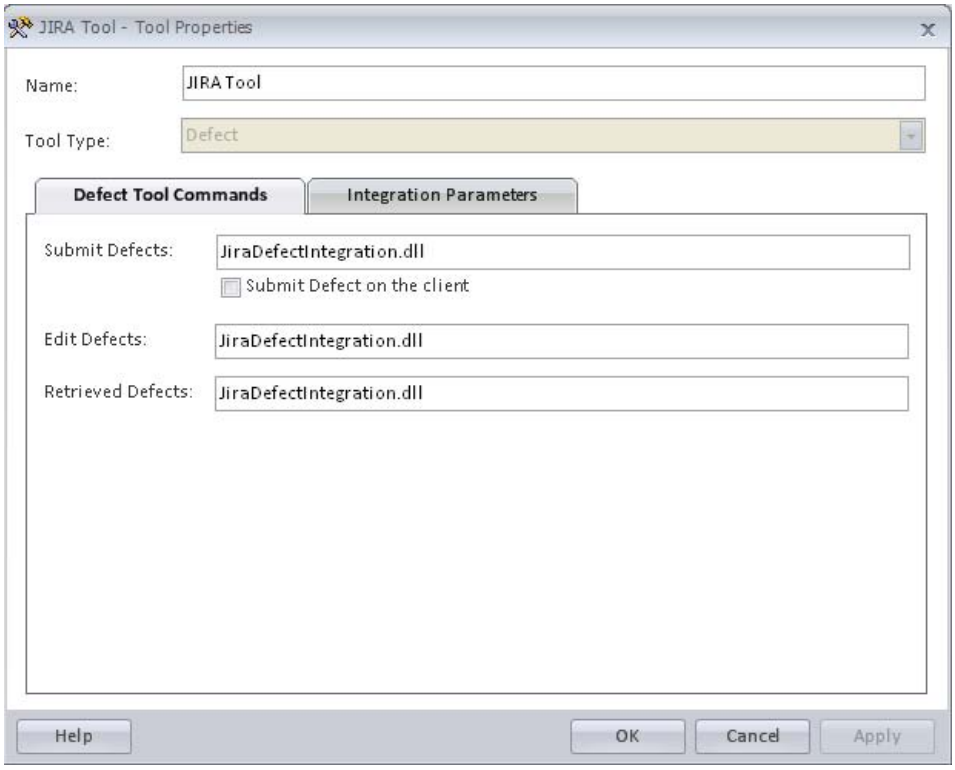
  Defect Tools
  Single Sign On is required for Defect submission and editing. They use the defect tool login specified in single sign on.

  For more information on Single Sign On and QADirector/third-party integrations, search for the following topic in the QADirector online help: *Integrating with External Products*.
- For Defect Tool integrations, be sure to associate the Project with the Tool Domain.

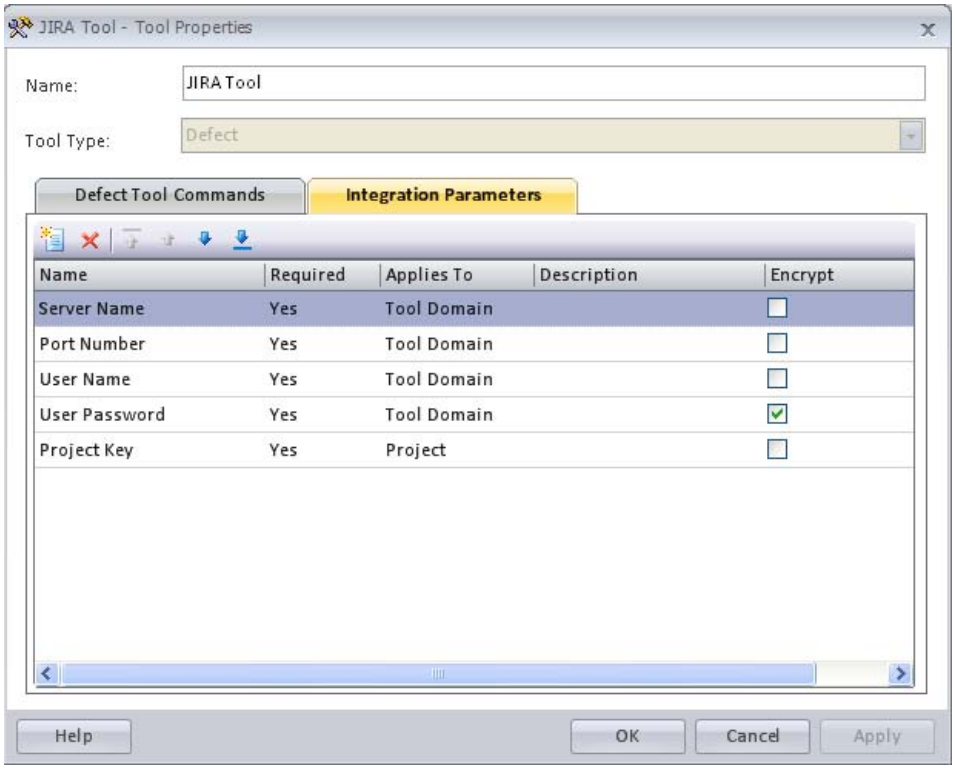
Setting Up a JIRA Integration

QADirector is shipped with integration assemblies used to integrate with the JIRA defect tracking tool. The following steps describe how to create the integration.

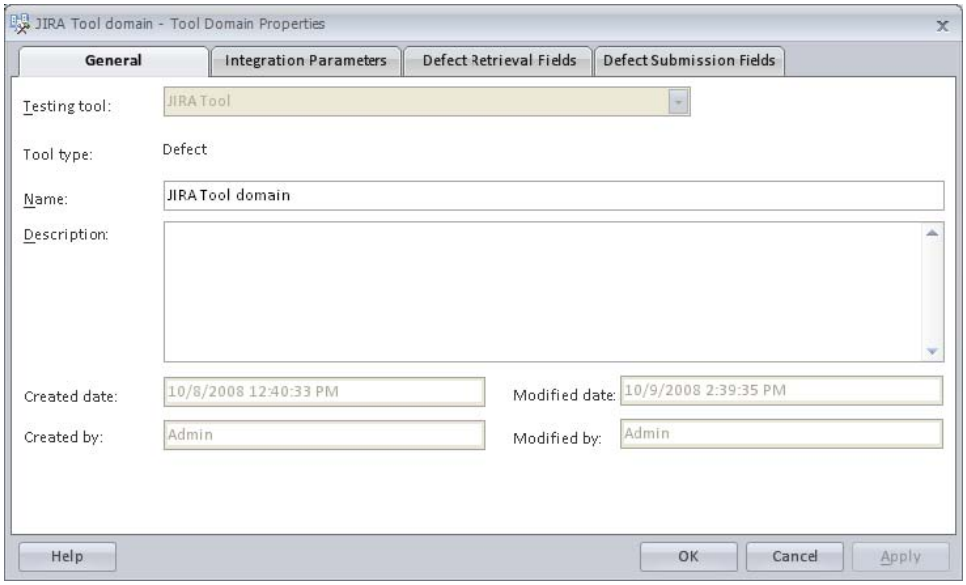
1. Copy the following JIRA integration assemblies from the source to destination directory:
   - Source: `\\QADirector SDK\Defect Tool Integration\JIRA\bin`
   - Destination: `\\Compuware\QADirector\TMServices\ThirdPartyIntegrations`
2. Create a JIRA tool by providing the appropriate defect tool commands (dll names) and Integration Parameters. Ensure that the Integration parameter names and settings are exactly as shown below. Select the **Defect Tool Commands** tab and enter the fields below.



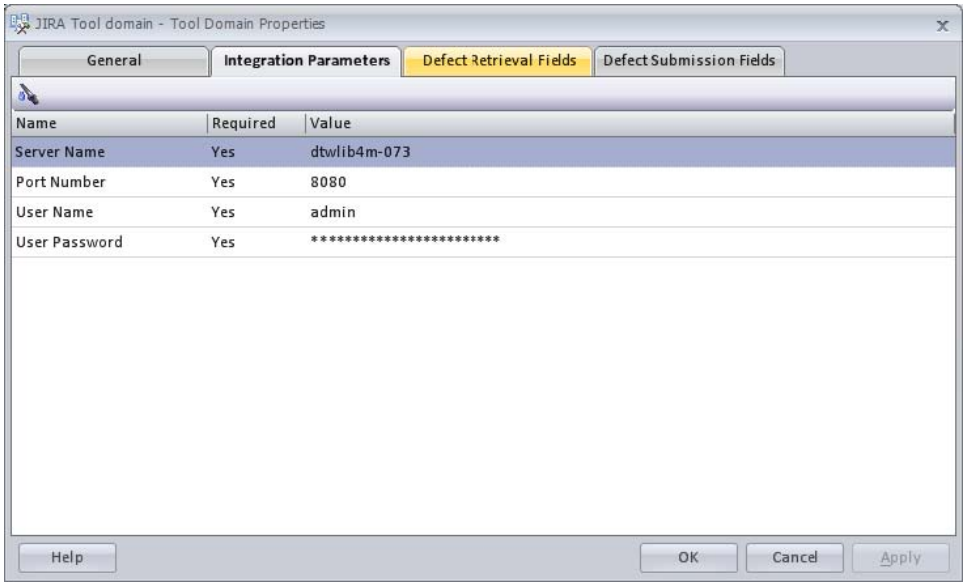
3. Select the **Integration Parameters** tab and enter the fields below:



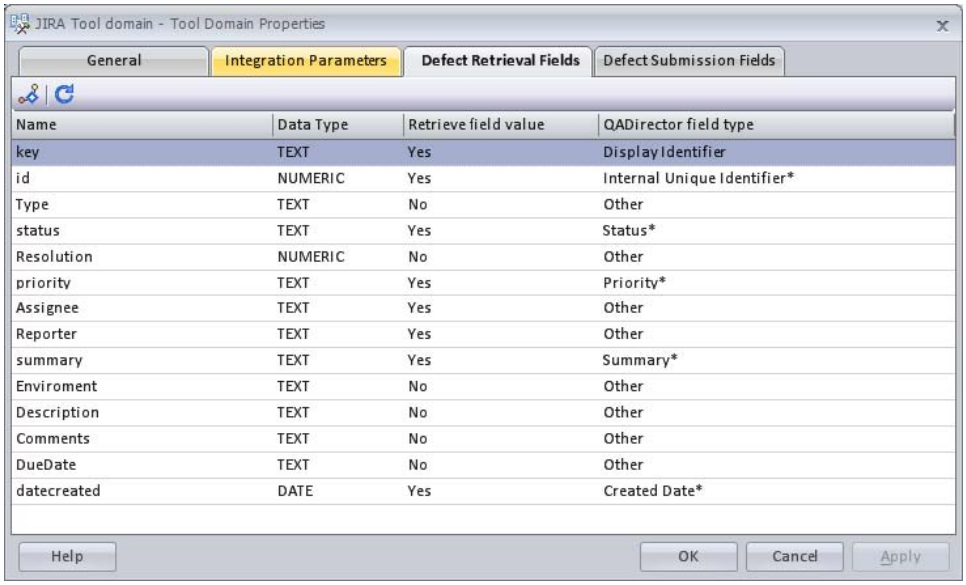
4. Create a new JIRA tool domain that belongs to the JIRA tool that was created in the previous step.



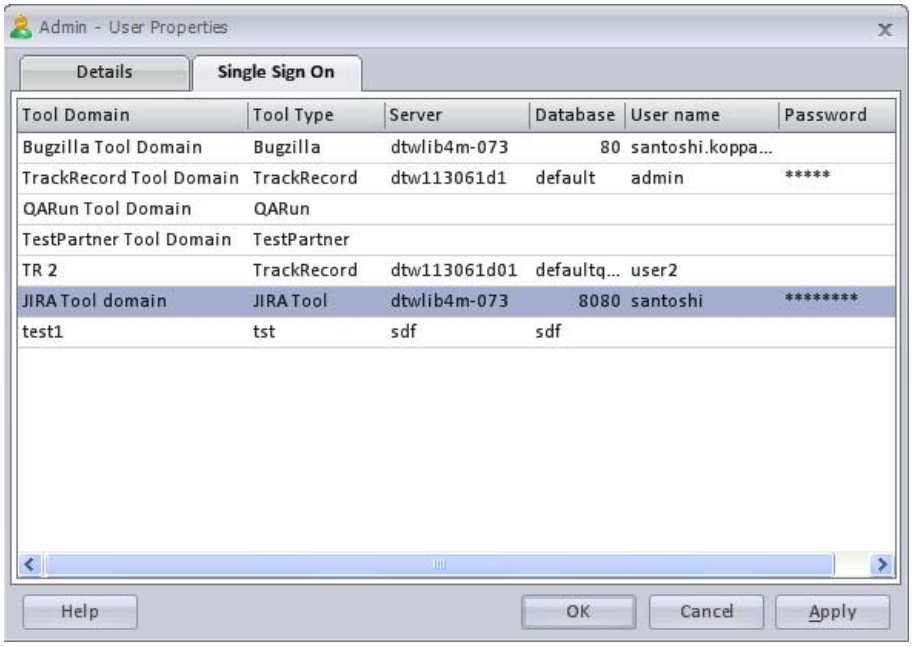
5. Select the **Integration Parameters** tab and provide the appropriate values for the integration parameters. For example, for the `Server Name` parameter, provide the JIRA server name. After providing the parameter values, save the tool domain and click the **Test tool domain** tool bar icon to test login to JIRA.



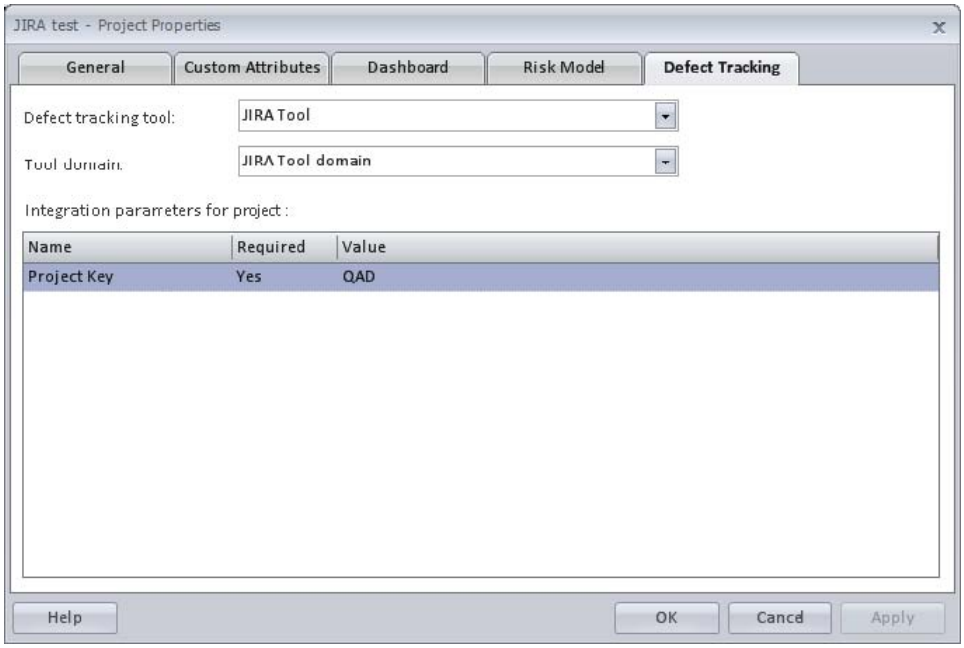
6. Select the **Defect Retrieval Fields** tab and click the **Reload** button to load JIRA's defect fields. In the **Retrieve field value** column, select **`Yes`** for the fields that you want to see in the QADirector **Defect Center**. Use the **Status Priority Mapping** tool bar icon to map priority and statuses (optional step). Make appropriate selections in the **Defect Submission** fields and save the tool domain.



7. Go to the **User Properties>Single Sign On** tab. Find the JIRA tool domain that was created in the previous step. For the JIRA tool domain, provide the JIRA login and password information.



8. Select the QADirector project of interest, go to **Project Properties>Defect Tracking** tab. Select the JIRA tool and tool domain, provide the JIRA project key to be used with the current QADirector project. Save the project properties.



This completes the JIRA integration set up. Defects can be submitted to QADirector or retrieved from JIRA to QADirector.

CHAPTER 3

Automated Tool Integration

QADirector exposes an integration interface which allows you to use the automated tool of your choice. Follow the steps in this section to use any third party automated tool to create, to edit, or to retrieve scripts. See Deployment [p. 51] after your code is compiled to complete the necessary steps of your integration.

The automated tool integration points include:

- Testing the connection to the tool.
- Retrieving field names from tool.
- Retrieving scripts from tool for display in QADirector **Global** and **Script Center.**
- Creating scripts from QADirector.
- Editing Scripts.

In order to complete the integration, you need to have technical knowledge of .NET-related technologies and be able to create a .NET application.

System Requirements

- An automated testing tool.
- Visual Studio and the .NET framework 2.0. (Compuware used Visual Studio 2005 for development and testing).
- Refer to the automated tool integration C# code samples installed with this SDK for a better understanding of the integration architecture.

Code Reference

Before building your custom DLL, analysis is needed to find the necessary APIs and integration points of the automated tool that will be used for QADirector integration. Per the QADirector

infrastructure, the integration requires a tool and tool domain set up with an external DLL that can be invoked by QADirector.

You will need to create a .NET application with an output type of *Class Library*. QADirector requires that the DLL has an interface named `IThirdPartyAutomated` and a class named `ToolClass`. Another interface named `IExecutionAPI` must be used in order to connect to the QADirector Test Execution Agent during script execution.

IThirdPartyAutomated Interface

The **`IThirdPartyAutomated Interface`** declares the methods that need to be implemented in the integration application.

See ToolClass [p. 42] for an example of how to create and implement this interface.

ToolClass

This class is derived from the IThirdPartyAutomated Interface [p. 42]. All of its interface methods should be implemented in this class.

```
interface IThirdPartyAutomated
{
   string GetScriptListFromTool(string inputXML);
   string EditScript(string inputXML);
   string NewScript(string inputXML);
   string GetScriptFieldListFromTool(string inputXML);
   bool   TestConnection(string inputXML);
   bool   RunScript();
}

ToolClass: IThirdPartyAutomated
{
  string GetScriptListFromTool(string inputXML);
  { //implementation}

  string EditScript(string inputXML);
  { //implementation}

  string NewScript(string inputXML);
  { //implementation}

  string GetScriptFieldListFromTool(string inputXML);
  { //implementation}

  bool   TestConnection(string inputXML);
  { //implementation}

  bool   RunScript();
  { //implementation}

 }
```

Parameter Information

The `inputXML` parameter sends the integration parameters and their values as defined in the tool and tool domain. Depending on which method is called, the inputXML may contain additional information.

Return Value

All element names in the return XML must be exactly as they appear in the examples in this section. For example, the element **`returncode`** cannot be **`ReturnCode.`**

Member Information

EditScript

Edits a script.

Syntax

`EditScript(string inputXML)`

Parameters

The input string is an XML string that is auto generated by QADirector based on the tool and tool domain settings. The XML will contain tool and tool domain values plus the properties of the selected script.

```
<root>
   <tool>
      <field name=\"Tool Attribute1\" value=\"this is my tool value\" />
   </tool>
   <tooldomain>
      <field name=\"Database Server Name\" value=\"dbservernamehere\" />
      <field name=\"Database Name\" value=\"dbnamehere\" />
      <field name=\"Database User Name\" value=\"dbusernamehere\" />
      <field name=\"Database User Password\" value=\"Oc+hog4CqPNRjIvHQeErAg==\" />
      <field name=\"Other value\" value=\"other value here\" />
   </tooldomain>
   <script name=\"script1\" description=\"script1Desc\">
      <field name=\"field1\" value=\"field1value\" />
      <field name=\"field2\" value=\"field2value\" />
   </script>
</root>
```

Type

`string`

GetScriptFieldListFromTool

Retrieves the fields information of the tool.

Syntax

`GetScriptFieldListFromTool(string inputXML)`

Parameters

The input string is an xml string that is auto generated by QADirector based on the tool and tool domain settings. The XML will contain only tool and tooldomain values.

```
<root>
   <tool>
      <field name=\"Tool Attribute1\" value=\"this is my tool value\" />
   </tool>
   <tooldomain>
      <field name=\"Database Server Name\" value=\"dbservernamehere\" />
      <field name=\"Database Name\" value=\"dbnamehere\" />
      <field name=\"Database User Name\" value=\"dbusernamehere\" />
      <field name=\"Database User Password\" value=\"Oc+hog4CqPNRjIvHQeErAg==\" />
      <field name=\"Other value\" value=\"other value here\" />
   </tooldomain>
</root>
```

Return Value

Returns a string field list from tool that will be selected or deselected to be viewed in the **Script Center**.

```
<root>
   <fields>
      <field name="Assignment" selected="0"  />
      <field name="Category"  selected="0" />
   </fields>
</root>
```

GetScriptListFromTool

Gets the Script list from the tool.

Syntax

`GetScriptListFromTool(string inputXML)`

Parameters

The input string is an xml string that is auto generated by QADirector based on the tool and tool domain settings. The XML will contain only tool and tooldomain values.

```
<root>
   <tool>
      <field name=\"Tool Attribute1\" value=\"this is my tool value\" />
   </tool>
   <tooldomain>
      <field name=\"Database Server Name\" value=\"dbservernamehere\" />
      <field name=\"Database Name\" value=\"dbnamehere\" />
      <field name=\"Database User Name\" value=\"dbusernamehere\" />
      <field name=\"Database User Password\" value=\"Oc+hog4CqPNRjIvHQeErAg==\" />
      <field name=\"Other value\" value=\"other value here\" />
```

```
   </tooldomain>
</root>
```

Return Value

XML string

```
<root>
   <script name="s1" description = "s1desc">
      <field name="This is my tool field1" value="This is my value" />
      <field name="This is my tool field2" value="This is my other value" />
   </>

   <script name="s2" description = "s2desc">
      <field name="This is my tool field1" value="This is my value" />
      <field name="This is my tool field2" value="This is my other value" />
   </>
</root>
```

NewScript

Creates a new script.

Syntax

```
NewScript(string inputXML)
```

Parameters

The input string is an XML string that is auto generated by QADirector based on the tool and tool domain settings. The XML will contain only tool and tooldomain values.

```
<root>
   <tool>
      <field name=\"Tool Attribute1\" value=\"this is my tool value\" />
   </tool>
   <tooldomain>
      <field name=\"Database Server Name\" value=\"dbservernamehere\" />
      <field name=\"Database Name\" value=\"dbnamehere\" />
      <field name=\"Database User Name\" value=\"dbusernamehere\" />
      <field name=\"Database User Password\" value=\"Oc+hog4CqPNRjIvHQeErAg==\" />
      <field name=\"Other value\" value=\"other value here\" />
   </tooldomain>
</root>
```

Return Value

string

RunScript

This method must be used in order to run a script. From within this method, a connection should be made to the QADirector Test Execution Agent using the IExecutionAPI Interface [p. 49] in order to call methods within the QADirector Test Execution Agent.

Syntax

```
RunScript()
```

Example

Use the following code to make a connection to the QADirector Test Execution Agent using the IExecutionAPI Interface [p. 49]:

```
IExecutionAPI eiAPI;
eiAPI = (IExecutionAPI)Activator.GetObject(typeof(IExecutionAPI),
"ipc://IntegrationCommunicationServer/IntegrationCommunicationServer.rem");
```

For more information, see Running a Script Example [p. 48].

TestConnection

Tests the connection to the tool by using the integration parameters provided in the tool domain.

Syntax

```
TestConnection (string inputXML)
```

Parameters

The input string is an xml string that is auto-generated by QADirector based on the tool and tool domain settings. The XML will contain only tool and tooldomain values.

```
<root>
   <tool>
      <field name=\"Tool Attribute1\" value=\"this is my tool value\" />
   </tool>
   <tooldomain>
      <field name=\"Database Server Name\" value=\"dbservernamehere\" />
      <field name=\"Database Name\" value=\"dbnamehere\" />
      <field name=\"Database User Name\" value=\"dbusernamehere\" />
      <field name=\"Database User Password\" value=\"Oc+hog4CqPNRjIvHQeErAg==\" />
      <field name=\"Other value\" value=\"other value here\" />
   </tooldomain>
</root>
```

Return Value

Returns a bool of True or False.

Getting Scripts Example

```
public string GetScriptListFromTool(string inputXML)
{
   //Build the xml. Input xml is expected to be in this format
   /*
   <root>
   <tool>
      <field name="This is my tool field1" value=”This is my value” />
      <field name="This is my tool field2" value=”This is my other value” />
   </tool>
   <tooldomain>
      <field name="This is my tool domain field1" value=”This is my value” />
      <field name="This is my tool domain field2" value=”This is my other value” />
   </tooldomain>
   <script>
      <field name="This is my script property field1" value=”This is my value” />
      <field name="This is my script property field2" value=”This is my other value” />

   </script>
   </root>
   /* return XML expected in this format
```

```
   <root>
      <script name=”s1” description = “s1desc”>
         <field name="This is my tool field1" value=”This is my value” />
         <field name="This is my tool field2" value=”This is my other value” /></>
      <script name=”s2” description = “s2desc”>
         <field name="This is my tool field1" value=”This is my value” />
         <field name="This is my tool field2" value=”This is my other value” /></>
   </root>
   */

   XmlDocument xmlDoc = new XmlDocument();
   XmlElement elMain = xmlDoc.CreateElement("root");
   xmlDoc.AppendChild(elMain);
   XmlElement newScriptElem;
   newScriptElem = this.CreateScriptNode(xmlDoc, "script1", "script1Desc");
   elMain.AppendChild(newScriptElem);
   XmlElement newFieldElem;
   newFieldElem = this.CreateFieldNode(xmlDoc, "f1", "feild1val");
   newScriptElem.AppendChild(newFieldElem);
   XmlElement newFieldElem2;
   newFieldElem2 = this.CreateFieldNode(xmlDoc, "f2", "feild1val2");
   newScriptElem.AppendChild(newFieldElem2);
   XmlElement newFieldElem3;
   newFieldElem3 = this.CreateFieldNode(xmlDoc, "f3", "feild1val2");
   newScriptElem.AppendChild(newFieldElem3);
   XmlElement newFieldElem4;
   newFieldElem4 = this.CreateFieldNode(xmlDoc, "f4", "feild1val2");
   newScriptElem.AppendChild(newFieldElem4);
   XmlElement newFieldElem5;
   newFieldElem5 = this.CreateFieldNode(xmlDoc, "f5", "feild1val2");
   newScriptElem.AppendChild(newFieldElem5);
   XmlElement newFieldElem6;
   newFieldElem6 = this.CreateFieldNode(xmlDoc, "f6", "feild1val2");
   newScriptElem.AppendChild(newFieldElem6);
   // script 2
   XmlElement newScriptElem1;
   newScriptElem1 = this.CreateScriptNode(xmlDoc, "script2", "script2Desc");
   elMain.AppendChild(newScriptElem1);
   XmlElement newFieldElem1;
   newFieldElem1 = this.CreateFieldNode(xmlDoc, "f1", "feild2val");
   newScriptElem1.AppendChild(newFieldElem1);
   XmlElement newFieldElem7;
   newFieldElem7 = this.CreateFieldNode(xmlDoc, "f2", "feild2val");
   newScriptElem1.AppendChild(newFieldElem7);
   return xmlDoc.InnerXml;
}
```

Getting Script Field List from Tool Example

```
public string GetScriptFieldListFromTool(string inputXML)
{

   /*<root>
   <tool>
      <field name="This is my tool field1" value=”This is my value” />
      <field name="This is my tool field2" value=”This is my other value” />
   </tool>
   <tooldomain>
      <field name="This is my tool domain field1" value=”This is my value” />
      <field name="This is my tool domain field2" value=”This is my other value” />
   </tooldomain>
      <script>
         <field name="This is my script property field1" value=”This is my value” />
        <field name="This is my script property field2" value=”This is my other value”
 />
      </script>
   </root>
   /* Return field list from tool that will be selected or deselected to be viewed in
Script Center
   <root>
      <fields>
         <field name="Assignment" selected="0" />
```

```
        <field name="Category" selected="0" />
      </fields>
   <root>
   inputXML gets the Tool Properties and Tool Domain
   */

   XmlDocument xmlDoc = new XmlDocument();
   XmlElement elMain = xmlDoc.CreateElement("root");
   xmlDoc.AppendChild(elMain);
   XmlElement newFieldsElem;
   newFieldsElem = xmlDoc.CreateElement("fields");
   elMain.AppendChild(newFieldsElem);
   XmlElement newFieldElem;
   newFieldElem = this.CreateRetrievalFieldNode(xmlDoc,"f1","0");
   newFieldsElem.AppendChild(newFieldElem);
   XmlElement newFieldElem1;
   newFieldElem1 = this.CreateRetrievalFieldNode(xmlDoc, "f2", "0");
   newFieldsElem.AppendChild(newFieldElem1);
   XmlElement newFieldElem2;
   newFieldElem2 = this.CreateRetrievalFieldNode(xmlDoc, "f3", "0");
   newFieldsElem.AppendChild(newFieldElem2);
   XmlElement newFieldElem3;
   newFieldElem3 = this.CreateRetrievalFieldNode(xmlDoc, "f4", "0");
   newFieldsElem.AppendChild(newFieldElem3);
   XmlElement newFieldElem4;
   newFieldElem4 = this.CreateRetrievalFieldNode(xmlDoc, "f5", "0");
   newFieldsElem.AppendChild(newFieldElem4);
   XmlElement newFieldElem5;
   newFieldElem5 = this.CreateRetrievalFieldNode(xmlDoc, "f6", "0");
   newFieldsElem.AppendChild(newFieldElem5);
   return xmlDoc.InnerXml;
```

Running a Script Example

The **`RunScript`** method must be used in order to run a script. From within this method, a connection should be made to the QADirector Test Execution Agent using the IExecutionAPI Interface [p. 49] in order to call methods within the QADirector Test Execution Agent.

```
public bool RunScript()
{
   string detailFileData = "";

   //Make a connection to the QADirector Test Execution Agent
   IExecutionAPI eiAPI;
   eiAPI = (IExecutionAPI)Activator.GetObject(typeof(IExecutionAPI),
"ipc://IntegrationCommunicationServer/IntegrationCommunicationServer.rem");

   //GetScriptParameters returns the Tool, Tool Domain, and current script definitions
   string inputXML = eiAPI.GetScriptParameters();

   /*
    * Script Execution logic here
    */

   //Example of passing a script
   bool bScriptStatus = true;

   if (bScriptStatus)
   {
      detailFileData = "This test passed.";
      eiAPI.SetResultOutcome(true);
   }
   else
   {
      detailFileData = "This test failed.";
      eiAPI.SetResultOutcome(false);
      eiAPI.SetResultString(detailFileData);
   }

   //Setting result file example
   System.Text.ASCIIEncoding encoding = new System.Text.ASCIIEncoding();
```

```
   byte[] fileBuffer = encoding.GetBytes(inputXML);

   eiAPI.SetResultFile("passed in data.xml", fileBuffer, false);

   //Setting detail result file example - Setting "isDetail" to true enables the Detail
 button
   //from the Script Result Properties dialog.
   fileBuffer = encoding.GetBytes(detailFileData);
   eiAPI.SetResultFile("detail.txt", fileBuffer, true);

   return true;
}
```

IExecutionAPI Interface

This interface must be used in order to pass information to the QADirector Test Execution Agent during script execution. It must be used in the definition of the RunScript [p. 45] method in the ToolClass [p. 42].

```
namespace Compuware.QACenter.QADirector.ExecutionInterface
{
   public interface IExecutionAPI
   {
      string GetScriptParameters();
      void SetResultFile(string fileName, byte[] resultFile, bool isDetail);
      void SetResultOutcome(bool resultOutcome);
      void SetResultString(string resultString);
   }
}
```

- For more information, see GetScriptParameters [p. 49].
- For more information, see SetResultFile [p. 50].
- For more information, see SetResultOutcome [p. 50].
- For more information, see SetResultString [p. 50].

GetScriptParameters

Gets information on the current script and its associated tool and tool domain.

Syntax

`GetScriptParameters()`

Return Value

The output string is an xml string that is auto generated by QADirector based on the tool and tool domain settings. The XML will contain tool and tool domain values plus the properties of the selected script.

```
<root>
   <tool>
      <field name=\"Tool Attribute1\" value=\"this is my tool value\" />
   </tool>
   <tooldomain>
      <field name=\"Database Server Name\" value=\"dbservernamehere\" />
      <field name=\"Database Name\" value=\"dbnamehere\" />
      <field name=\"Database User Name\" value=\"dbusernamehere\" />
      <field name=\"Database User Password\" value=\"Oc+hog4CqPNRjIvHQeErAg==\" />
      <field name=\"Other value\" value=\"other value here\" />
   </tooldomain>
```

```
   <script name=\"script1\" description=\"script1Desc\">
      <field name=\"field1\" value=\"field1value\" />
      <field name=\"field2\" value=\"field2value\" />
   </script>
</root>
```

SetResultFile

Used to pass a file back to the QADirector Test Execution Agent which can then be viewed within QADirector's results.

Syntax

```
SetResultFile(string fileName, byte[] resultFile, bool isDetail)
```

Parameters

- `string fileName` - The name of the file. This file name will appear within QADirector results for this particular script.
- `byte[] resultFile` - A byte array that represents the file's contents.
- `bool isDetail` - If this is set to true, when the results are viewed in QADirector's **Job Results Detail**, if this script is double-clicked within the user interface, it will try to launch this file. Only one detail file can be set per script per execution. When set to false, this file will appear on the **Result Summary** window in QADirector **Job Results Detail**.

Return Value

`void`

SetResultOutcome

Used to pass or fail the current script that is running.

Syntax

```
SetResultOutcome(bool resultOutcome)
```

Parameters

`bool resultOutcome` - Set to true to pass the script. Set to false to fail the script.

Return Value

`void`

SetResultString

Used to pass a failure message in the event that a script fails. This should be used in conjunction with SetResultOutcome [p. 50], where the result outcome was a failed script. The failure message will appear in QADirector's **Job Result Detail**.

Syntax

```
SetResultString(string resultString)
```

Parameters

`string resultString` - The failure message for the script.

Return Value

`void`

Deployment

1. After coding the integration DLL, the output binaries should be copied to the following location on the QADirector web server machine: `\Compuware\QADirector\TMServices\ThirdPartyIntegrations`. Your DLL will be deployed in the following manner:

   Automated Tools

   Your DLL will be downloaded from the server to the client when needed.

   Defect Tools

   - For defect retrieval, the DLL is executed on the server.
   - For defect editing, the DLL is always downloaded and executed on the client on demand.
   - For defect submission, the default behavior is silent submission on the server. However, the **Tool Properties** dialog box provides an option to submit defects on the client. If this option is selected, then the DLL will be download and executed on the client.

2. Create a **Tool** and **Tool Domain** in QADirector to allow QADirector to send and receive the appropriate information.

   - When creating a **Tool**, be sure to select the appropriate type from the **Tool Type** list. For example: for defects, select the **`Defect`** type. For automated tools, select the **`Automated`** type.
   - The **Tool/Tool Domain** integration parameters are created and configured based on the needs of the tool/integration:

     Automated Tools

     Parameters can be set to apply at the tool domain or the tool properties. By applying a parameter at the tool properties, a value can be set that can be used in the tool. Additional Parameters can be added.

     Defect Tools

     Parameters for defect tool domains can be set to apply at the tool domain or the project level. By applying a parameter at the project level, duplicate tool domains can be avoided. The integration parameter values are set in two locations.

Parameters that apply across the tool domain are set in the tool domain properties integration parameters tab, while parameters that apply to projects are set in the project properties defect tracking tab.

- **Single Sign On**

  Automated Tools
  You can optionally use **Single Sign On** for the integration.

  Defect Tools
  Single Sign On is required for Defect submission and editing. They use the defect tool login specified in single sign on.

  For more information on Single Sign On and QADirector/third-party integrations, search for the following topic in the QADirector online help: *Integrating with External Products*.

- For Defect Tool integrations, be sure to associate the Project with the Tool Domain.

CHAPTER 4

Requirements Management

The requirements management tool integration is used to integrate a requirements management tool with QADirector. Requirements are imported into the QADirector database as Requirement and Test assets. Properties of these assets (Results, ID, etc) can then be exported into requirement management tools.

System Requirements

System requirements for the requirements management tool integration include:

- QADirector API (TMClient.dll).
- Requirements management tool.
- Visual Studio 2005/.NET 2.0.
- Refer to the RM integration C# code samples installed with this SDK for a better understanding of the integration architecture.

Code Reference

Client Class

Provides access to all of the information about a given `Compuware.QM.QADirector.SDK.Client` in QADirector.

Member of: `Compuware.QM.QADirector.SDK`

Methods

- CloseClient [p. 54]
- OpenClient [p. 55]

Properties

CloseClient

Closes a Client.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Client`

Syntax

```
CloseClient()
```

CustomAttributes

Gets a collection of all **`CustomAttributes`** associated with the **`Client`**.

Type: `Compuware.QM.QADirector.SDK.CustomAttributes`

Member of: `Compuware.QM.QADirector.SDK.Client`

Description

Gets the **`Client`** description.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Client`

ID

Gets the **`Client`** ID.

Type: `int`

Member of: `Compuware.QM.QADirector.SDK.Client`

ManualTestFolders

Gets a collection of all **`ManualTestFolders`** associated with the **`Client`.**

Type: `Compuware.QM.QADirector.SDK.ManualTestFolders`

Member of: `Compuware.QM.QADirector.SDK.Client`

Name

Gets the **`Client`** name.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Client`

OpenClient

Opens a client.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Client`

Syntax

```
OpenClient()
```

Projects

Gets a collection of all **`Project`** objects associated with the **`Client`**.

Type: `Compuware.QM.QADirector.SDK.Projects`

Member of: `Compuware.QM.QADirector.SDK.Client`

RiskModels

Gets a collection of all the **`RiskModel`** objects associated with the **`Client`**

Type: `Compuware.QM.QADirector.SDK.RiskModels`

Member of: `Compuware.QM.QADirector.SDK.Client`

TestingTools

Gets a collection of all **`Tool`** objects associated with the **`Client`**.

Type: `Compuware.QM.QADirector.SDK.Tools`

Member of: `Compuware.QM.QADirector.SDK.Client`

Users

Gets a collection of all **`Users`** objects associated with the **`Client`**.

Type: `Compuware.QM.QADirector.SDK.Users`

Member of: `Compuware.QM.QADirector.SDK.Client`

Clients Class

Provides access to all of the `Clients` in QADirector.

Member of: `Compuware.QM.QADirector.SDK`

Properties

- Count [p. 56]
- Clients [p. 56]

Clients

- **`Clients[int clientid]`** gets a **`Client`** object by the supplied clientid.
- **`Clients[string clientname]`** gets a **`Client`** object by the supplied name.

Type: `Compuware.QM.QADirector.SDK.Client`

Count

Gets the count of clients.

Type: `int`

Member of: `Compuware.QM.QADirector.SDK.Clients`

Connection Class

Connects to **Test Management web services**. Users needs to retain a reference to access other exposed functionality.

Member of: `Compuware.QM.QADirector.SDK`

Methods

- IsConnected [p. 57]
- LogOn [p. 58]
- LogOff [p. 58]

Properties

- APIVersion [p. 57]
- Clients [p. 57]
- CurrentClient [p. 57]
- CurrentProject [p. 57]
- CurrentUser [p. 57]
- Roles [p. 58]

- Users [p. 58]

Code Sample

Connection/Logon Example [p. 59]

APIVersion

Gets the QADirector API version.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Connection`

Clients

Gets a collection of **`Clients`** objects.

Type: `Compuware.QM.QADirector.SDK.Clients`

Member of: `Compuware.QM.QADirector.SDK.Connection`

CurrentClient

Gets a **`Client`** object of the currently opened client.

Type: `Compuware.QM.QADirector.SDK.Client`

Member of: `Compuware.QM.QADirector.SDK.Connection`

CurrentProject

Gets a **`Project`** object of the currently opened project.

Type: `Compuware.QM.QADirector.SDK.Project`

Member of: `Compuware.QM.QADirector.SDK.Connection`

CurrentUser

Gets a **`User`** object of the current logged in user.

Type: `Compuware.QM.QADirector.SDK.User`

Member of: `Compuware.QM.QADirector.SDK.Connection`

IsConnected

Indicates if the connection is still active.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Connection`

Syntax

```
IsConnected()
```

LogOff

Logs off a user.

Type: `void`

Member of: `Compuware.QM.QADirector.SDK.Connection`

Syntax

LogOff()

LogOn

Authenticates a user to log on to the SDK.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Connection`

Syntax

```
LogOn(string UserName,string Password, string serverName,string
virtualDirectory, bool isSSL)
```

Parameters

- `string UserName` - User name to connect to QADirector.
- `string Password` - Password to connect to QADirector.
- `string serverName` - Name of the web server to connect to.
- `string virtualDirectory` - Name of the Virtual directory.
- `bool isSSL` - Indicates if SSL is enabled on the server.

Overloads

`LogOn` is an overloaded method. If your web server's default `Port Number` is not 80, indicate the port number using the overloaded method.

`Int PortNumber` - Web server's Port number.

Roles

Gets the collection of **`Roles`** objects.

Type: `Compuware.QM.QADirector.SDK.Roles`

Member of: `Compuware.QM.QADirector.SDK.Connection`

Users

Gets the collection of **`Users`** objects.

Type: `Compuware.QM.QADirector.SDK.Users`

Member of: `Compuware.QM.QADirector.SDK.Connection`

Connection/Logon Example

```
bool IsAuth = con.Logon(UserName, Password, WebSiteName, VirtualDir, WebSitePort,isSSL);
if (con.IsConnected())
{
   //set the current user
   curUser = con.CurrentUser;
}
else
{
 //problem logging in
}
```

ManualSteps Class

Use this class to add steps to a manual script.

Member of: `Compuware.QM.QADirector.SDK`

Methods

New [p. 59]

New

This method adds a new Step to a **Manual Script**.

Type: `ManualSteps`

Member of: `Compuware.QM.QADirector.SDK.ManualSteps`

Syntax

```
New(string scriptname, string step, int orderno, int steptype,string
choices,string correctanswer, string expectedresult,string associateddata)
```

Parameters

- `string scriptname` is the name of the Script.
- `string step` is the text of the Step.
- `int orderno` is the order number of the Step
- `int steptype` is a **`StepType`** enumeration:
  - **`NONE = 0`**
  - **`QUESTIONTEXT = 1`**
  - **`MULTIPLECHOICE = 2`**
  - **`INSTRUCTION = 3`**
  - **`QUESTION = 7`**
- `string choices` is a list of choices separated by the **`^`** character. For example:

- **`Pass^fail`**
- **`True^false`**

- `string correctanswer` is one of the choices that is supposed to be the correct answer.
- `string expectedresult` is description of the expected result.
- `string associateddata` is the description of any other information to add to the step.

Nodes Class

The `Compuware.QM.QADirector.SDK.Nodes` class keeps an ordered collection of `Node` objects. The hierarchy is maintained in this collection.

Member of: `Compuware.QM.QADirector.SDK`

Methods

- NewRMReqNode [p. 60]
- NewRMTestNode [p. 61]

Properties

- Count [p. 60]

Code Sample

Building an RM Node Collection Example [p. 61]

Count

Gets node count inclusive of all Tests and Requirements.

Type: `int`

Member of: `Compuware.QM.QADirector.SDK.Nodes`

NewRMReqNode

Creates a new Requirement `Node` object.

Type: `Compuware.QM.QADirector.SDK.Node`

Member of: `Compuware.QM.QADirector.SDK.Nodes`

Syntax

```
NewRMReqNode(string ExternalID)
```

Parameters

`string ExternalID` is the External ID of the third party Requirement.

NewRMTestNode

Creates a new `Node` of type *Test*.

Type: `Compuware.QM.QADirector.SDK.Node`

Member of: `Compuware.QM.QADirector.SDK.Nodes`

Syntax

`NewRMTestNode(string ExternalID)`

Parameters

`string ExternalID` is the External ID of the third party Requirement.

Building an RM Node Collection Example

```
Node root, r1, r2, t1, t2;

rmFolder = curProject.RMFolder;
nodes = rmFolder.Nodes;

LastAlphaRequirementID = "";

root = nodes.NewRMReqNode(GetNextAlphaID());
root.Name = "Req 1";
root.Description = "Req 1 descrip";

#region Level 2
r1 = root.Nodes.NewRMReqNode(GetNextAlphaID());
r1.Name = "Req 1.1";
r1.Description = "MY Req 1.1 descrip";
#endregion

r2 = root.Nodes.NewRMReqNode(GetNextAlphaID());
r2.Name = "Req 1.2";
r2.Description = "MY Req 1.2 descrip";

#region Level 4 - Tests
t1 = r1.Nodes.NewRMTestNode(GetNextAlphaID());
t1.Name = "Test1";
t1.Description = "MY Test1 descrip";
#endregion

//code to import data into application
rmFolder.Connection.CurrentProject.RMToolName = RMToolNames.CaliberRM;
rmFolder.Connection.CurrentProject.RMProjectName = "Caliber's Test Proj";
bool blnVal = rmFolder.CreateRMTree();

//code to update data in QADirector
RMUpdateOpts ropts = new RMUpdateOpts();
ropts.getResult = (int)eRMResultChoice.All;
ropts.getRisk = false;
bool blnVal = rmFolder.UpdateRMTree(ropts);
```

Project Class

Provides access to all the information for a given project in QADirector.

Member of: `Compuware.QM.QADirector.SDK`

Methods

Properties

AddScript

Adds a global script to a project.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Project`

Syntax

```
AddScript(Script srpt,int iFolderID)
```

Parameters

- `Script srpt` is the `Compuware.QM.QADirector.SDK.Script` object.
- `int iFolderID` is the Id of the folder/tool domain to which the script belongs.

CloseProject

Closes a **`Project`**.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Project`

Syntax

```
CloseProject()
```

Description

Gets/sets the project description.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Project`

EndDate

Gets the end date of the project.

Type: `System.DateTime`

Member of: `Compuware.QM.QADirector.SDK.Project`

ExecutionPlans

Gets the collection of **`ExecutionPlans`** associated with the project.

Type: `Compuware.QM.QADirector.SDK.ExecutionPlans`

Member of: `Compuware.QM.QADirector.SDK.Project`

GetScript

Returns a **`Script`** object from the Project Scripts collection.

Type: `Compuware.QM.QADirector.SDK.Script`

Member of: `Compuware.QM.QADirector.SDK.Project`

Syntax

```
GetScript(int ScriptDefnID)
```

Parameters

`int ScriptDefnID` is the ID of the Script to return.

ID

Gets the ID of the project.

Type: `int`

Member of: `Compuware.QM.QADirector.SDK.Project`

Name

Gets the project name.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Project`

NumberOfCycles

Gets/sets the number of cycles associated with the project.

Type: `int`

Member of: `Compuware.QM.QADirector.SDK.Project`

OpenProject

Opens a project.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Project`

Syntax

```
OpenProject()
```

RMFolder

Gets the **`RMFolder`** associated with the project.

Type: `Compuware.QM.QADirector.SDK.RMFolder`

Member of: `Compuware.QM.QADirector.SDK.Project`

RMIntegrated

Gets a boolean indicating if the current project is integrated with RM.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Project`

RMKey

Gets the database key of the RM project.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Project`

RMProjectID

Gets/sets the RM project id.

Type: `int`

Member of: `Compuware.QM.QADirector.SDK.Project`

RMProjectName

Gets/sets the RM project name.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Project`

RMToolName

Gets/sets the tool name of the RM project.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Project`

RequirementFolders

Gets the collection of **`RequirementFolders`** associated with the project.

Type: `Compuware.QM.QADirector.SDK.RequirementFolders`

Member of: `Compuware.QM.QADirector.SDK.Project`

ResetRMIntegration

Resets the integration with the requirements management application.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Project`

Syntax

```
ResetRMIntegration()
```

ResultFolders

Gets the collection of **`ResultFolders`** associated with the project.

Type: `Compuware.QM.QADirector.SDK.ResultFolders`

Member of: `Compuware.QM.QADirector.SDK.Project`

RiskModelID

Gets the risk model ID associated with the project.

Type: `int`

Member of: `Compuware.QM.QADirector.SDK.Project`

Scripts

Gets the collection of **`Scripts`** associated with the project.

Type: `Compuware.QM.QADirector.SDK.Scripts`

Member of: `Compuware.QM.QADirector.SDK.Project`

StartDate

Gets the Start date of the project.

Type: `System.DateTime`

Member of: `Compuware.QM.QADirector.SDK.Project`

TestFolders

Gets the collection of **`TestFolder`** objects associated with the project.

Type: `Compuware.QM.QADirector.SDK.TestFolders`

Member of: `Compuware.QM.QADirector.SDK.Project`

Tests

Gets all of the tests associated with the project.

Type: `Compuware.QM.QADirector.SDK.Tests`

Member of: `Compuware.QM.QADirector.SDK.Project`

Update

Updates the project information to the repository.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Project`

Syntax

```
Update()
```

Projects Class

Provides access to all of the projects in a given QADirector Client.

Member of: `Compuware.QM.QADirector.SDK`

Methods

Properties

Count

Gets the count of projects.

Type: `int`

Member of: `Compuware.QM.QADirector.SDK.Projects`

Exists

Validates the existence of a project.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Projects`

Syntax

```
Exists(string projectname)
```

Parameters

`String projectname` is the name of the project.

New

Creates a new project with the given name and description.

Type: `Compuware.QM.QADirector.SDK.Project`

Member of: `Compuware.QM.QADirector.SDK.Projects`

Syntax

```
New(ProjectName, ProjectDescription)
```

Parameters

- `String projectname` - Name of the project.
- `String projectdescription` - Description of the project.

Projects

Syntax

- **`Projects[int projectid]`** - Returns a **`Project`** object by project id via integer.
- **`Projects[string projectname]`** - Returns a **`Project`** object by project name via string.

Refresh

Refreshes the **`Projects`** collection with the latest list of projects from database.

Type: `void`

Member of: `Compuware.QM.QADirector.SDK.Projects`

Syntax

```
Refresh()
```

RMFolder Class

RMFolder is a key class for requirements management integration. All of the data push/pull is done through this class.

Member of: `Compuware.QM.QADirector.SDK`

Methods

- CreateRMTree [p. 68]
- ResetRMIntegration [p. 70]
- RMNewEP [p. 69]
- RMReplaceEP [p. 69]
- RMUpdateEP [p. 69]
- UpdateRMTree [p. 70]

CreateRMTree

Creates the Requirement/Test hierarchy in the application under **Default Requirement Folder**.

Pre-Condition: The nodes hierarchy is already built in `RMFolder`.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.RMFolder`

Syntax

```
CreateRMTree()
```

RMNewEP

Creates a new **ExecutionPlan** with the given name. This takes all the requirements in the **MasterFolder** and creates a new **ExecutionPlan**.

Type: `Compuware.QM.QADirector.SDK.ExecutionPlan`

Member of: `Compuware.QM.QADirector.SDK.RMFolder`

Syntax

```
RMNewEP(string EPName, bool bCreateGroups)
```

Parameters

`string EPName` is the name of the **ExecutionPlan.**

`bool bCreateGroups` creates/ignores groups while creating the **ExecutionPlan**.

RMReplaceEP

Updates an existing **ExecutionPlan** by removing all the nodes in it and recreating the hierarchy from the master folder.

Type: `Compuware.QM.QADirector.SDK.ExecutionPlan`

Member of: `Compuware.QM.QADirector.SDK.RMFolder`

Syntax

```
RMReplaceEP(int EPDefnID, string EPName, bool bCreateGroups)
```

Parameters

- `int EPDefnID` is the ID of the **ExecutionPlan**.
- `string EPName` is the name of the **ExecutionPlan.**
- `bool bCreateGroups` creates/ignores groups while creating an **ExecutionPlan.**

RMUpdateEP

Updates an existing **ExecutionPlan** by removing all the nodes in it and recreating the hierarchy from the master folder.

Type: `Compuware.QM.QADirector.SDK.ExecutionPlan`

Member of: `Compuware.QM.QADirector.SDK.RMFolder`

Syntax

```
RMUpdateEP(int EPDefnID, string EPName, bool bCreateGroups)
```

Parameters

- `Int EPDefnID` is the ID of the Execution plan.
- `String EPName` is the Name of the Execution plan.
- `Bool bCreateGroups` creates/ignores groups while creating the Execution plan.

ResetRMIntegration

Resets the integration with the QADirector database.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.RMFolder`

Syntax

```
ResetRMIntegration()
```

UpdateRMTree

Updates the **DefaultRequirement** folder in the application with the nodes content of **`RMFolder.`**

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.RMFolder`

Syntax

```
UpdateRMTree(RMUpdateOpts upOpts)
```

Parameters

`RMUpdateOpts` specifies what options need to be brought back as part of export:

1. `int getResult`
2. `Int32 resultFolderID`
3. `bool getRisk`

Script Class

Provides access to a script in a given project in QADirector.

Member of: `Compuware.QM.QADirector.SDK`

Methods

- Update [p. 71]

Properties

- CreatedByUserName [p. 71]
- Description [p. 71]
- ID [p. 71]
- ModifiedByUserName [p. 71]
- Name [p. 71]

CreatedByUserName

Gets the user who created the script.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Script`

Description

Gets/sets the description of the **`Script.`**

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Script`

ID

Gets the ID of the script.

Type: `int`

Member of: `Compuware.QM.QADirector.SDK.Script`

ModifiedByUserName

Gets the user who modified the script.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Script`

Name

This property returns the name of the script.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Script`

Update

This method updates a script with the new name and description.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Script`

Syntax

```
Update()
```

Scripts Class

Provides access to all the scripts in a given project in QADirector.

Member of: `Compuware.QM.QADirector.SDK`

Methods

Properties

Count

Gets the number of scripts in a project.

Type: `int`

Member of: `Compuware.QM.QADirector.SDK.Scripts`

Delete

This method deletes a script with the given id.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Scripts`

Syntax

```
Delete(int scriptdefnid)
```

Parameters

`int scriptdefnid` is the ID of the Script to delete.

Exists

This method checks for the existence of a script by either the script name or script id, depending on which version f the method used.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Scripts`

Syntax

- **`Exists(string ScriptName)`**
- **`Exists[int Scriptdefnid]`**

GetScript

This method returns a **`Script`** object for the specified script id.

Type: `Compuware.QM.QADirector.SDK.Script`

Member of: `Compuware.QM.QADirector.SDK.Scripts`

Syntax

```
GetScript(int ScriptDefnID)
```

Parameters

`int ScriptDefnID` is the ID of the **`Script.`**

New

This method creates a new **`Script`** object.

Type: `Compuware.QM.QADirector.SDK.Script`

Member of: `Compuware.QM.QADirector.SDK.Scripts`

Syntax

```
New()
```

Overloads

```
New(string name, string description)
```

Refresh

This method refreshes the **`Scripts`** collection from the database.

Type: `void`

Member of: `Compuware.QM.QADirector.SDK.Scripts`

Syntax

`Refresh()`

Return Value

Void

Script

Gets a **`Script`** object.

Syntax

- **`Script[string scriptname]`**
- **`Script[int scriptdefnid]`**

Returns

`Compuware.QM.QADirector.SDK.Script`

Test Class

Provides access to all of the information of a test in a given project in QADirector.

Member of: `Compuware.QM.QADirector.SDK`

Methods

Properties

- ScriptCount [p. 77]
- ScriptNodes [p. 77]
- Status [p. 77]
- Testdefnid [p. 77]

AddScript

This method associates a Script with a Test.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Test`

Syntax

```
AddScript(Script srpt)
```

Parameters

`Script srpt` is a `Compuware.QM.QADirector.SDK.Script` object.

AssignedTo

Gets the name of the user to whom the test is assigned.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Test`

CreatedByUserName

Gets the name of the user who created the test.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Test`

DefectCount

Gets the count of defects associated with the test.

Type: `int`

Member of: `Compuware.QM.QADirector.SDK.Test`

Description

Gets the description of the test in a string.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Test`

LastResult

Gets the last result of the test in a string.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Test`

ModifiedByUserName

Gets the name of the user who modified the test.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Test`

Name

Gets the name of the test.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Test`

RemoveScript

This method removes the association between the test and a script.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Test`

Syntax

```
RemoveScript(Script srpt)
```

Parameters

`Script srpt` is a `Compuware.QM.QADirector.SDK.Script` object.

RequirementCount

Gets the count of requirements associated to the test.

Type: `int`

Member of: `Compuware.QM.QADirector.SDK.Test`

Risk

Gets the risk of a test.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Test`

ScriptCount

This property returns the count of scripts associated to the test.

Type: `int`

Member of: `Compuware.QM.QADirector.SDK.Test`

ScriptNodes

Gets an ArrayList of scripts associated with the test.

Type: `System.Collections.ArrayList`

Member of: `Compuware.QM.QADirector.SDK.Test`

Status

Gets the status of the test.

Type: `string`

Member of: `Compuware.QM.QADirector.SDK.Test`

Testdefnid

Gets the id of the test.

Type: `int`

Member of: `Compuware.QM.QADirector.SDK.Test`

Update

Updates a **`Test`** with all the associated scripts.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Test`

Syntax

```
Update()
```

UpdateLight

This method updates a test with the new name and description.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Test`

Syntax

```
UpdateLight()
```

Tests Class

Provides access to all the tests in a given project in QADirector.

Member of: `Compuware.QM.QADirector.SDK`

Methods

Properties

Count

Gets the number of tests in a project.

Type: `int`

Member of: `Compuware.QM.QADirector.SDK.Tests`

Delete

Deletes one or more tests.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Tests`

Syntax

```
Delete(string testids)
```

Parameters

`string testids` is a comma separated string of testids.

Exists

Checks for the existence of a **`Test`** with a given `Name`.

Type: `bool`

Member of: `Compuware.QM.QADirector.SDK.Tests`

Syntax

```
Exists(string TestName)
```

Parameters

`string TestName` is the name of a Test.

GetAsset

Returns a `Compuware.QM.QADirector.SDK.Test` object for the specified test id.

Type: `Compuware.QM.QADirector.SDK.Test`

Member of: `Compuware.QM.QADirector.SDK.Tests`

Syntax

```
GetAsset(int AssetDefnID)
```

Parameters

`int AssetDefnID` is the asset ID.

New

Creates a new **`Test`** object.

Type: `Compuware.QM.QADirector.SDK.Test`

Member of: `Compuware.QM.QADirector.SDK.Tests`

Syntax

```
New()
```

Refresh

Refreshes the **`Tests`** collection in the database.

Type: `void`

Member of: `Compuware.QM.QADirector.SDK.Tests`

Syntax

```
Refresh()
```

Test

Gets a **`Test`** object with either the supplied asset name or asset id.

Syntax

- **`Test[int assetdefnid]`** returns a **`Test`** object by `assetdefnid`
- **`Test[string assetname]`** returns a **`Test`** object by test `assetname`.

Type: `Compuware.QM.QADirector.SDK.Test`

Member of: `Compuware.QM.QADirector.SDK.Tests`

Deployment

Be sure to deploy QADirector's `tmclient.dll` and `localization.dll` with your application.

Index

M

N

O

P

R

S

T

U