Enterprise Integration Kit Manual

for Ultimus Adaptive BPM Suite 8.3 SP2

Copyright information

This document was last updated on June 3, 2013.

Contents

Overview

Chapter 1
EIK interface description

Chapter 2
Events Subscription Interface

Chapter 3
Ultimus BIService Web service

Chapter 4
Ultimus EIKService Web service

Chapter 5
Ultimus ClientServices Web service

Chapter 6
Process level Web service methods

Chapter 7
Initiating business process incidents from third-party applications

Chapter 8
The Ultimus Custom OC interface

Chapter 9
Customizing Ultimus BPM Server e-mail messages

Chapter 10
The Ultimus Form Control Object

Overview

Introduction

The Ultimus Adaptive BPM Suite *Enterprise Integration Kit* (EIK) *Manual* provides detailed technical information how to customize and extend the functionality of the Ultimus Adaptive BPM Suite. Ultimus BPM Server provides an open, extensible environment for business process automation. It can be used as a fully functional and complete out-of-the-box solution for business process automation, or it can be embedded in third-party software packages to provide BPM functionality behind the scenes. Between these two extremes, it allows different levels of integration with third-party applications.

This manual is designed for software developers and Ultimus Adaptive BPM Suite advanced users.

Contacting Ultimus

Ultimus is always striving to improve its product and support services. Furthermore, Ultimus offers a number of ways to find answers or to submit feedback.

You may use the following ways to find answers to your Ultimus-related questions or to submit feedback to Ultimus:

- **Ultimus Customer Portal:** At the Ultimus Customer Portal, you can access technical experts to resolve your technical issues, use the KnowledgeBase to get answers to common and specific questions, and download the latest product builds and documentation. You can reach Ultimus Support at: `https://www.ultimussupport.com/`.
- **Ultimus Education:** Ultimus Education provides technical training and certification on the latest Ultimus BPM Suite to ensure you possess up-to-date knowledge of the latest product releases. Ultimus Enterprise Integration Kit (EIK) training is provided on an as-needed basis. For more information, contact `training@ultimus.com`.
- **Product enhancement:** Ultimus strives to improve our product. If you would like to submit a product enhancement or feature concept, go to the Ultimus Customer Portal at: `https://www.ultimussupport.com` and follow the “Ideas” link in the upper area.
- **Documentation feedback:** Ultimus strives to improve technical documentation and online help. If you would like to submit documentation feedback, contact `documents@ultimus.com`.

Conventions used in this document

The following conventions are used throughout this document:

bold	Bold text denotes items that you must select or click on in an application, such as menu options, dialog box options, and dialog box output. Bold text is also used to designate labels within table columns.
italic	Italic text denotes variables, emphasis, and document, chapter, and section titles. This also denotes text that is a place holder for a word or value that you must supply.
`monospace`	Text in this font denotes text or characters that you should input to an application, application output, sections of code, programming examples, and syntax examples. This is also used for the proper names of disk drives, paths, directories, device names, file names, file extensions, code excerpts, and hyperlinks.
`monospace italic`	Italic text in this font denotes text that is a placeholder for text or value(s) that you must supply.
»	The » symbol leads you through nested Start menu options, application menu options, and dialog box options to a final action. For example, the sequence **File»Page Setup»Printer...** directs you to pull down the **File** menu, select the **Page Setup** item, then select **Printer...** from the dialog box.
	This icon denotes a tip, which alerts you to advisory information.
	This icon denotes a note, which alerts you to important information.
	This icon denotes a caution, which advises you of precautions to take to avoid specific application errors, data loss, or system crash.
	This icon denotes a warning, which advises you of precautions to take to avoid damaging computer hardware or losing computer data.

Prerequisites

Before using the Ultimus EIK or its documentation, verify the following:

- Ultimus BPM Server 8.3 SP2 must be installed somewhere in the Ultimus BPM environment.
- The logged on user must have extensive knowledge of Ultimus Adaptive BPM Suite 8.3 SP2.

- This document assumes the reader has expertise in the following software technologies:
  - Ultimus Adaptive BPM Suite 8.3 SP2
  - Microsoft C# or VB.NET
  - Visual C++
  - ASP.NET
  - Web services/XML
  - ActiveX, JavaScript, and DHTML
  - COM+ (and an understanding of how to develop a COM component in any programming language
  - Basic database concepts

Supported and unsupported use of the Ultimus EIK

This section outlines official supported and unsupported use of the Ultimus EIK.

Supported use

Ultimus supports the Ultimus EIK under the following guidelines and in accordance with the Ultimus EIK purchase agreement:

a. All interfaces, classes, and methods outlined in this document are supported for use in custom implementations which are developed in coding languages as sanctioned in this document.

b. Custom solutions built upon the EIK are supported only to the extent in which the Ultimus EIK interfaces, classes, and methods are supported. It is the responsibility of the customer to isolate the EIK issue within the custom solution to the specific lines of code causing the problem and submit the isolated code to Ultimus for problem analysis.

c. Support for custom solutions may be possible if the customer has a Premier Solution Support Agreement. For further information, please contact your regional account manager or sales representative.

Unsupported use

Ultimus does not support the use of the Ultimus EIK under the following guidelines:

a. Ultimus will not support a custom solution if the customer does not have a Premier Solution Support Agreement. For further information, please contact your regional account manager or sales representative.

b. Usage of any Ultimus Adaptive BPM Suite DLLs, EXEs, or other components in custom solutions, where the DLLs or EXEs are not discussed in the Ultimus EIK Manual, is not supported. Ultimus does not support undocumented functionality or features.

c. Custom solutions that directly read from Ultimus BPM Database tables are not supported.

d. Custom solutions that directly write to Ultimus BPM Database tables are not supported.

e. Custom solutions that affect Ultimus Adaptive BPM Suite functionality in ways not discussed in the Ultimus EIK Manual are not supported.

Outline of this document

The Ultimus Adaptive BPM Suite *Enterprise Integration Kit Manual* is broken down into sections. Each section pertains to a distinct description of methods or functionality within the Ultimus EIK. Below is an outline of each section.

EIK description

The following sections outline method descriptions within the .NET EIK assembly that are designed to be called within the Microsoft development environment:

- *EIK interface description*: The `UltEIK.dll` and `UltEIkStructures.dll` files contain all of the .NET classes necessary to manage Ultimus business processes. This section outlines all method and property descriptions within the `UltEIK.dll` and `UltEIkStructures.dll` files.
- *Events Subscription Interface*: This section describes how event systems function in general, as well as how an event subscription interface functions within the Ultimus EIK.

Web services

The following sections outlines descriptions and examples of how to implement Web services within the Ultimus EIK:

- *Ultimus BIService Web service*: The Ultimus BIService Web service is an interface for business process and task data to third-party applications. The third-party reporting applications can leverage the Ultimus BIService Web service to retrieve business and process level data to produce quick and data-rich dashboards, portals, and reports.
- *Ultimus ClientServices Web service*: The Ultimus ClientServices API is new with Ultimus Adaptive BPM Suite 8. This interface is not only utilized by Ultimus Client and Ultimus Thin Client, but also should be utilized when building custom clients.
- *Process level Web service methods*: Ultimus Adaptive BPM Suite provides a variety of Web service methods. Any tool or application that is capable of consuming WSDL 1.1- and SOAP 1.1-compliant Web services can leverage these methods.

Launching business processes

The following section outlines method descriptions for using third-party applications to launch business processes:

- *Initiating business process incidents from third-party applications*: With the techniques outlined in this section, any third-party application can complete the Begin step of a business process.

Extending Ultimus BPM Server

The following sections outline how to extend Ultimus BPM Server's functionality. These include how to utilize a Custom OC interface and how to customize e-mails sent from Ultimus BPM Server. These include:

- *The Ultimus Custom OC interface*: For organizations already using third-party applications or a customized database to manage organizational hierarchy information, the Ultimus OC (Organization Charts) Translation Layer interface provides the option of synchronizing the Ultimus Adaptive BPM Suite with any existing third-party application. This section describes benefits and considerations in using the Custom OC interface as well as an outline of method descriptions it uses.
- *Customizing Ultimus BPM Server e-mail messages*: By default, Ultimus BPM Server integrates with Microsoft Exchange Server in order to process and deliver e-mail notifications and messages to Ultimus Adaptive BPM Suite users. To extend Ultimus e-mail functionality, Ultimus offers its users the ability to integrate the Ultimus Adaptive BPM Suite with any third-party e-mail system. This section describes how to customize Ultimus BPM Server e-mail messages through any third-party e-mail system.

Extending Ultimus forms

The following sections outline how to extend the functionality using Ultimus Forms. These include utilizing the Ultimus Form Control Object (FCO) and using JavaScript in forms. These include:

- *The Ultimus Form Control Object*: The Form Control Object (FCO) is a .Net User control that is hosted in Internet Explorer. The FCO controls the data transfer between the controls in a step's Ultimus Form and its corresponding step XML schema within a business process.
- *Using JavaScript in Ultimus Forms*: Using JavaScript in Ultimus Forms lets the process designer modify the functionality of the forms according to process requirements. This adds great flexibility to Ultimus Adaptive BPM Suite functionality.

Error status codes returned by Flobots outlines error status codes returned by Flobots that are installed with Ultimus Adaptive BPM Suite 8.3 SP2.

1

EIK interface description

The Ultimus EIK (Enterprise Integration Kit) .NET assembly is a dynamic link library that is designed to be called as a resource in the Microsoft development environment. The `UltEIK.dll` file contains all of the .NET classes necessary to manage Ultimus business processes. The following Microsoft components are required to use the Ultimus EIK:

- Any of the following server-class operating systems:
  a. Microsoft Windows Server 2003
  b. Microsoft Windows Server 2003 R2
  c. Microsoft Windows Server 2008
  d. Microsoft Windows Server 2008 R2
- Microsoft .NET Framework 2.0
- Microsoft Visual Studio 2005, where the custom application calling the Ultimus EIK is built in VB.NET, C#, or ASP.NET

Note: Custom applications built using unmanaged code (such as VB, C++, or ASP) which call the Ultimus EIK are not supported.

As part of Ultimus BPM Server's installation, the `UltEIK.dll` and `UltEIkStructures.dll` files are installed in the global assembly cache (GAC). The `UltEIK.dll` file is always installed on the computer hosting the Ultimus BPM Server. The assembly is discussed in detail in this section. Some custom solutions might not be able to reference files in the GAC when compiled or executed. As such, the `UltEIK.dll` file can be copied from the folder where Ultimus BPM Server is installed to the `/bin` folder associated with the customer solution source files.

Caution: Due to the increasing sophistication of the product, Ultimus does not support direct customer interaction with the Ultimus BPM Database. Instead, Ultimus supports data connectivity through the Ultimus EIK. While a broad array of methods are provided, Ultimus encourages customers and partners to request new EIK methods as a Product Enhancement through the Ultimus Customer Portal at `https://www.ultimussupport.com/` and follow the "Ideas" link in the upper area.

Documentation for the EIK interface description contains the following sections:

- *The Ultimus EIK interface: an overview*
- *The Ultimus EIK interface: classes within the* `WFServer` *namespace*
- *The Ultimus EIK interface: classes within the* `OC` *namespace*
- *The Ultimus EIK interface: classes within the* `Configuration` *namespace*
- *Using the Ultimus EIK with Ultimus-context information*

The Ultimus EIK interface: an overview

The Ultimus EIK interface is composed of three namespaces (shown in Figure 1 on page 24):

- `Ultimus.WFServer`
- `Ultimus.OC`
- `Ultimus.Configuration`

Within each of these namespaces, there are a number of classes. The word “ENUM” next to particular classes represent classes which are enumerations.

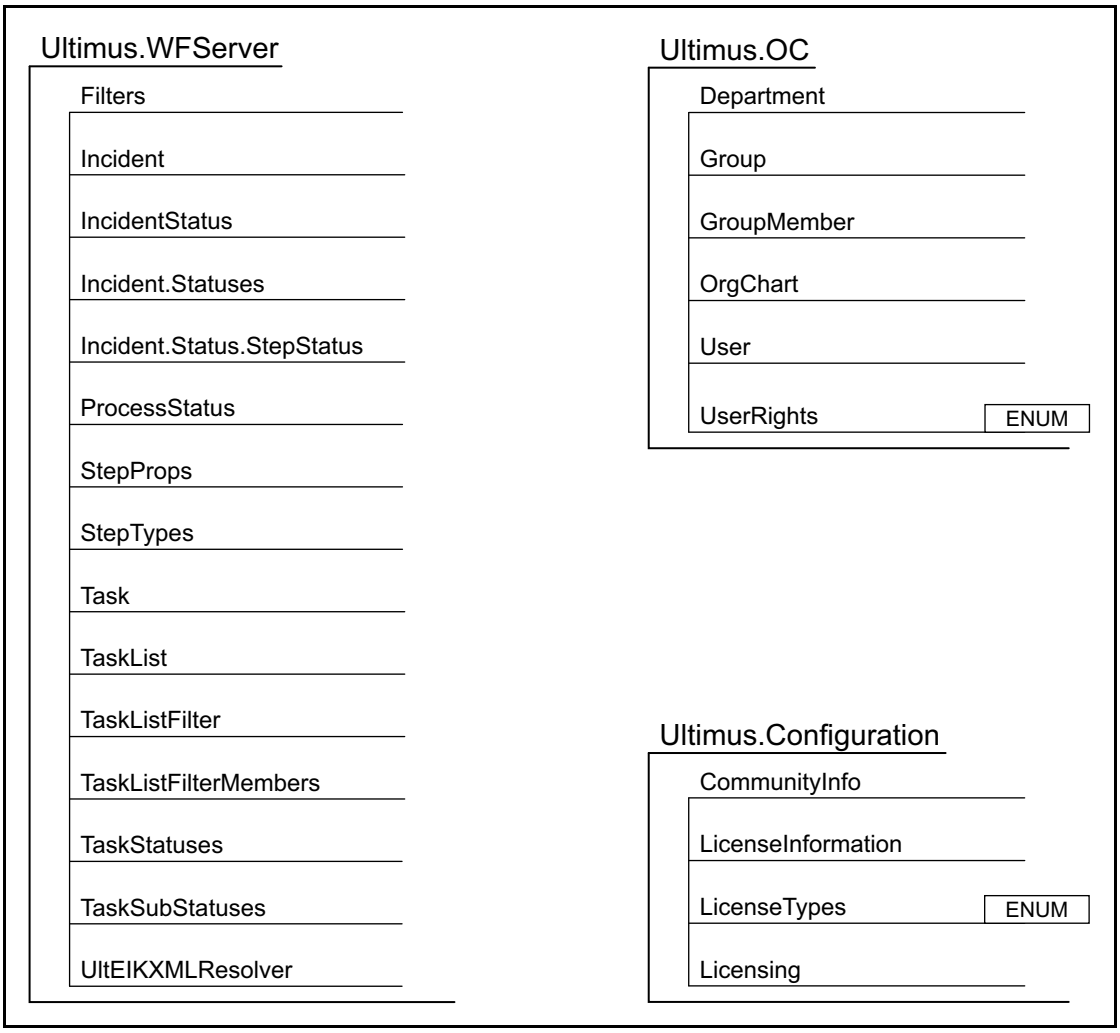

Figure 1. The Ultimus EIK class structure

The `WFServer` namespace contains 15 classes (shown in Figure 2).

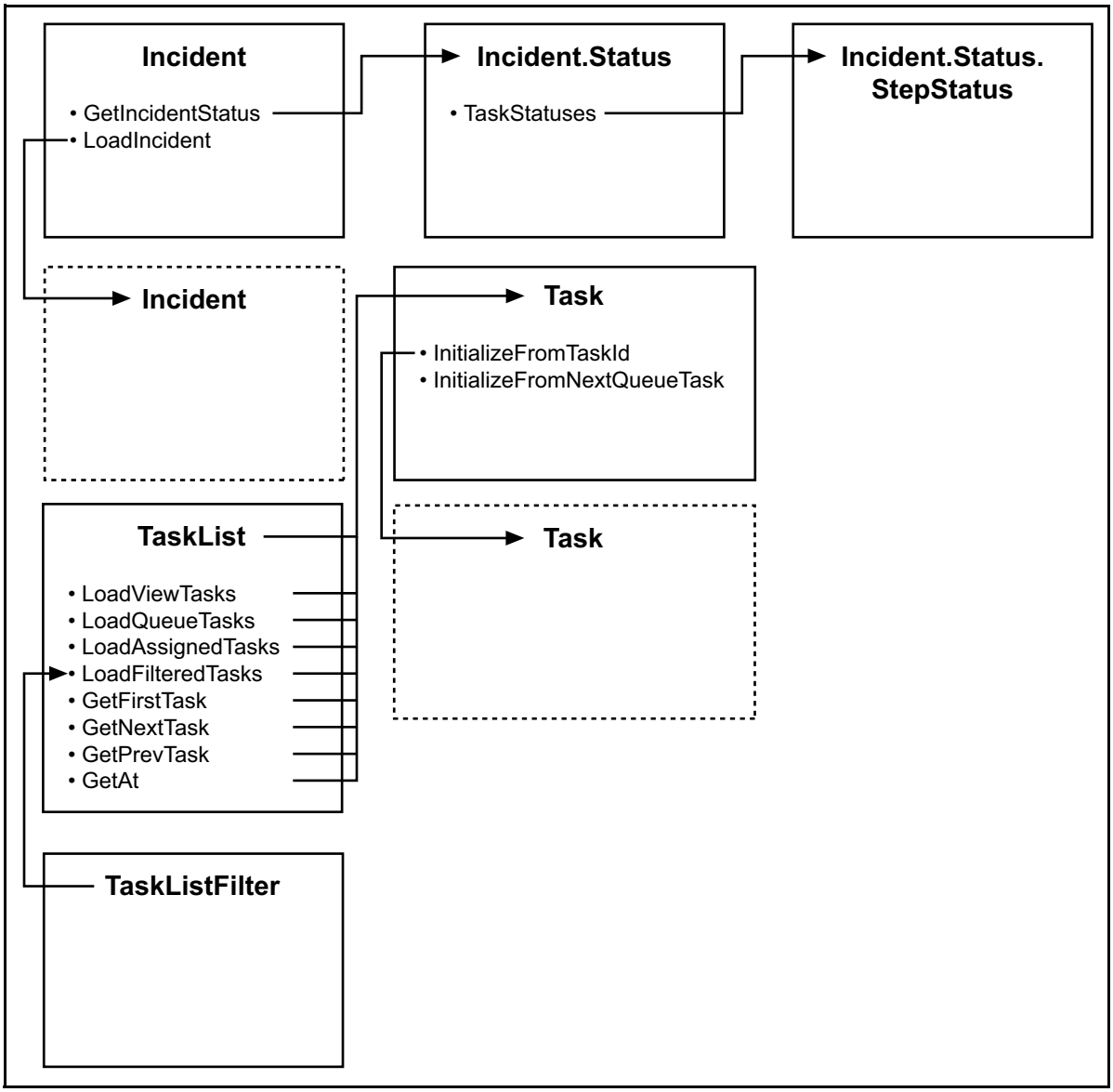

Figure 2. An architectural schematic for the `WFServer` namespace

Each class contains methods and properties (shown in Figure 3).

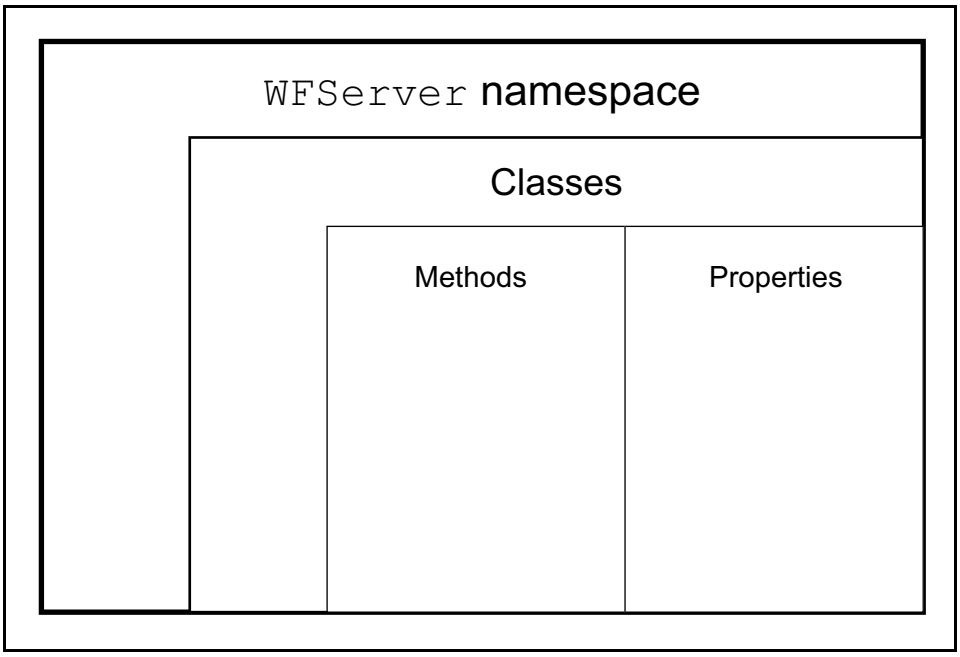

Figure 3. The Ultimus EIK architectural relationship

The Ultimus EIK interface: classes within the `WFServer` namespace

Outlined in the tables below are descriptions of all status codes, properties, and methods within each class of the `WFServer` namespace. This section contains:

- *Enumerations (ENUMs)*
- *Properties and methods in the `Incident` class*
- *Properties and method in the `Incident.Status` class*
- *Properties in the `Incident.Status.StepStatus` class*
- *Properties and methods in the `Task` class*
- *Methods in the `TaskList` class*
- *Properties and methods in the `TaskListFilter` class*

Enumerations (ENUMs)

The following table outlines status codes pertaining to filters. These are defined in `WFServer.Filters` class of the `ulteikstructures.dll` file.

Table 1. Enumerations for `WFServer.Filters` class

Enumerations	Description
`nFilter_Initiate`	Begin step task
`nFilter_Urgent`	Urgent task
`nFilter_Overdue`	Overdue task
`nFilter_Current`	Current task (active task that is not urgent or overdue)
`nFilter_Complete`	Completed task
`nFilter_Assigned`	Assigned task
`nFilter_IncidentComplete`	Completed incident
`nFilter_Queue`	Queue task

The following table outlines status codes pertaining to the status of incidents. These are defined in `WFServer.IncidentStatuses` class of the `ulteikstructures.dll` file.

Table 2. Enumerations for `WFServer.IncidentStatuses` class

Enumerations	Description
`INCIDENT_STATUS_ABORTED`	Aborted incident
`INCIDENT_STATUS_ACTIVE`	Active incident
`INCIDENT_STATUS_COMPLETED`	Completed incident
`INCIDENT_STATUS_PENDING`[1]	Pending incident
`INCIDENT_STATUS_STALLED`	Stalled incident

1. For the `INCIDENT_STATUS_PENDING`, an Ultimus incident temporarily has a status of PENDING after the Begin step of the incident is completed and before the next step is activated. Once the next step in the incident is activated, the incident status changes from PENDING to ACTIVE.

The following table outlines status codes pertaining to the status of published business processes. These are defined in `WFServer.ProcessStatus` class of the `ulteikstructures.dll` file.

Table 3. Enumerations for `WFServer.ProcessStatus` class

Enumeration	Description
`PROCESS_STATUS_ACTIVE`	Active process
`PROCESS_STATUS_DISABLED`	Disabled process

The following table outlines status codes pertaining to step properties. These are defined in `WFServer.StepProps` class of the `ulteikstructures.dll` file.

Table 4. Enumerations for `WFServer.StepProps` class

Enumeration	Description
`STEP_PROPERTY_PRIVATE`	Private (not visible to others)
`STEP_PROPERTY_PROTECTED`	Protected (not assignable to others)
`STEP_PROPERTY_NOTIFYTASKOWNER`	Notify task owner (in case late)
`STEP_PROPERTY_NOTIFYSUPERVISOR`	Notify supervisor (in case late)
`STEP_PROPERTY_DOARCHIVE`	Archive (yes/no)
`STEP_PROPERTY_PERPETUAL`	Resubmittable (yes/no)
`STEP_PROPERTY_DISABLED`	Disable (process can't be launched)
`TASK_PROPERTY_TEMPLATE`	Task is template

The following table outlines status codes pertaining to step types. These are defined in `WFServer.StepTypes` class.

Table 5. Enumerations for `WFServer.StepTypes` class

Enumeration	Description
`STEP_TYPE_BEGIN`	Begin step
`STEP_TYPE_END`	End step
`STEP_TYPE_USER`	User step
`STEP_TYPE_FLOBOT`	Flobot step

Table 5. Enumerations for `WFServer.StepTypes` class *(Continued)*

Enumeration	Description
STEP_TYPE_PROCESS	Maplet step
STEP_TYPE_JUNCTION	Junction step

The following table outlines status codes pertaining to task list filter members. These are defined in `WFServer.TaskListFilterMembers` class of the `ulteikstructures.dll` file.

Table 6. Enumerations for `WFServer.TaskListFilterMembers` class

Enumeration	Description
TLFILTER_MEMBER_DUEDATES	This is used to sort a task list based on overdue tasks (date/time).
TLFILTER_MEMBER_INCIDENT_SUMMARY	This is used to sort tasks on the basis of the incident summary.
TLFILTER_MEMBER_INCIDENTS	This is used to sort tasks on the incident number.
TLFILTER_MEMBER_PRIORITIES	This is used to sort tasks on the basis of the Ultimus priority setting.
TLFILTER_MEMBER_PROCESSNAME	This is used to sort tasks on the basis of the process name.
TLFILTER_MEMBER_SIMPLESTATUS	This is used to sort tasks on the basis of client views.
TLFILTER_MEMBER_STEPLABELS	This is used to sort tasks on the basis of step label text.
TLFILTER_MEMBER_TASKSTATUS	This is used to sort tasks on the basis of task status.
TLFILTER_MEMBER_TASKSUBSTATUS	This is used to sort tasks on the basis of task sub-status.
TLFILTER_MEMBER_USERS	This is used to sort task on the basis of user name.

The following table outlines status codes pertaining to task statuses. These are defined in `WFServer.TaskStatuses` class of the `ulteikstructures.dll` file.

Table 7. Enumerations for `WFServer.TaskStatuses` class

Enumeration	Description
TASK_STATUS_ABORTED	Task aborted
TASK_STATUS_ACTIVE	Task is active
TASK_STATUS_COMPLETED	Task complete

Table 7. Enumerations for `WFServer.TaskStatuses` class *(Continued)*

Enumeration	Description
`TASK_STATUS_DELAYED`	Task delayed
`TASK_STATUS_FAILED`	Task has failed
`TASK_STATUS_INQUEUE`	Task is in queue
`TASK_STATUS_REJECTED`	Task rejected
`TASK_STATUS_SKIPPED`	Task skipped

The following table outlines status codes pertaining to completed task sub-statuses. These are defined in `WFServer.TaskSubStatuses` class of the `ulteikstructures.dll` file.

Table 8. Enumerations for `WFServer.TaskSubStatuses` class

Enumeration	Description
`TASK_STATUS_ACTIVE_CONFERED`	Task is active and conferred
`TASK_STATUS_ACTIVE_LATE`	Task is active but late
`TASK_STATUS_ACTIVE_OVERDUE`	Task is active but overdue
`TASK_STATUS_ACTIVE_REASSIGNED`	Task is active and was reassigned
`TASK_STATUS_ACTIVE_RETURNED`	Task is active and returned
`TASK_STATUS_COMPLETE_INCIDENT_COMPLETE`	Task has successfully completed, and the incident has completed

Properties and methods in the `Incident` class

The `Incident` class represents the incident of an Ultimus process. All interactions with an incident take place within this class. Outlined below are the properties and methods for the `Incident` class (in alphabetical order):

- *Properties for the `Incident` class*
- *AbortIncident*
- *AbortStep*
- *AddMemoEntry*
- *AppendToRepeatedNode*
- *DeleteIncident*

- *DirectActivateStep*
- *DirectActivateStepEx*
- *GetIncidentStatus*
- *GetIncidentXML*
- *GetMapVersion*
- *GetMemo*
- *GetNodeValue*
- *GetProcessSchema*
- *GetRepeatedNodeSize*
- *GetStepStatus*
- *InsertIntoRepeatedNode*
- *LoadIncident*
- *RemoveRepeatedNodeIndex*
- *SaveVariables*
- *SetIncidentXML*
- *SetNodeValue*

Properties for the `Incident` class

Table 9. Properties for the `Incident` class

Property Name	Property Type and Description
`nIncidentNo`	**Type:** System.Int32 **Description:** `nIncidentNo` provides a unique identifier for this incident of an Ultimus process.
`nVersion`	**Type:** System.Int32 **Description:** `nVersion` corresponds with the version of the process to which this incident is associated.
`strIncidentOwner`	**Type:** System.String **Description:** `strIncidentOwner` corresponds with the name of the person who initiated this incident.

Table 9. Properties for the `Incident` class *(Continued)*

Property Name	Property Type and Description
`strIncidentSummary`	**Type:** System.String **Description:** `strIncidentSummary` corresponds with a current incident summary as entered by the user.
`strProcessName`	**Type:** System.String **Description:** `strProcessName` corresponds with the name of the process to which this incident is associated.

AbortIncident

Method: AbortIncident	

Method description: This method aborts the actively loaded incident.
C# method call:
`public bool AbortIncident(string strUserName, string strReason, out string strError)`
VB.NET method call:
`Public Function AbortIncident(ByVal strUserName As String, ByVal strReason As String, ByRef strError As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` is the short user name (or log on ID) of a user who is aborting the incident of an Ultimus process. The ID of the user who completed the Begin step of the incident should be passed for this parameter.
`strReason`	**Input Type:** System.String **Description:** `strReason` is a string offering a reason for aborting this incident of an Ultimus process.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to abort the incident.
Return Conditions	
`AbortIncident` returns `TRUE` if the incident could be aborted. `AbortIncident` returns `FALSE` if the incident could not be aborted.	

AbortStep

Method: AbortStep	

Method description: This method aborts a specified step. This method has the same effect of `Task.AbortStep` method.
 	
C# method call:
`public bool AbortStep (string strStepLabel, string strStepID)`
 	
VB.NET method call:
`Public Function AbortStep(ByVal strStepLabel As String, ByVal strStepID As String) As Boolean`	

Input	
`strStepID`	**Input Type:** System.String **Description:** `strStepID` contains the step ID which is to be aborted.
`strStepLabel`	**Input Type:** System.String **Description:** `strStepLabel` contains the step name which is to be aborted.
Return Conditions	
`AbortStep` returns `TRUE` if AbortStep successfully aborts the task. `AbortStep` returns `FALSE` if AbortStep does not successfully abort the task.	
Example	

```
'This code snippet is written in VB.NET
Dim oIncident As New Ultimus.WFServer.Incident
oIncident.LoadIncident("processname", incidentno)
If oIncident.AbortStep("Step Name", "Step ID") Then
   MessageBox.Show("Step Aborted Sucessfully")
End If
```

AddMemoEntry

Method: AddMemoEntry	

Method description: This method adds an incident memo to a specified step.
 	
C# method call:
`public bool AddMemoEntry(string strUser, string strStepLabel, string strMemoEntry, out string strError)`
 	
VB.NET method call:
`Public Function AddMemoEntry(ByVal strUser As String, ByVal strStepLabel As String, ByVal strMemoEntry As String, ByRef strError As String) As Boolean`	

Input	
`strUser`	**Input Type:** System.String **Description:** `strUser` represents the name of the user against which a memo entry is made.
`strStepLabel`	**Input Type:** System.String **Description:** `strStepLabel` represents the name of the step for which the memo is to be updated.
`strMemoEntry`	**Input Type:** System.String **Description:** `strMemoEntry` represents the text to be added to the memo.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to add a memo.
Return Conditions	
`AddMemoEntry` returns `TRUE` if an entry is made successfully. `AddMemoEntry` returns `FALSE` if memo is not added successfully.	

Method: AddMemoEntry
Example

```
'This code snippet is written in VB.NET
Dim oIncident As New Incident
Dim strError As String = ""

oIncident.LoadIncident("processname", incidentno)
If oIncident.AddMemoEntry("UserName", "StepName", "Memo details", strError) Then
   MessageBox.Show("memo is added successfully")
Else
   MessageBox.Show(strError)
End If
```

AppendToRepeatedNode

Method: AppendToRepeatedNode	

Method description: This method appends an additional node to a specified repeated incident XML schema element.
 	
C# method call:
`public bool AppendToRepeatedNode(string strRepeatedNodePath, out string strError)`
 	
VB.NET method call:
`Public Function AppendToRepeatedNode(ByVal strRepeatedNodePath As String, ByRef strError As String) As Boolean`	

Input	
`strRepeatedNodePath`	**Input Type:** System.String **Description:** `strRepeatedNodePath` represents the path of the repeated node where an additional node is to be appended.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to append to a repeated node.

Method: AppendToRepeatedNode
Return Conditions
`AppendToRepeatedNode` returns `TRUE` if the node is appended successfully. `AppendToRepeatedNode` returns `FALSE` if the node is not appended successfully.
Example

```
'This code snippet is written in VB.NET
Dim oIncident As New Ultimus.WFServer.Incident
Dim strNodeName As String = "IncidentData.Global.NodeName"
Dim strError As String = ""

oIncident.LoadIncident("processname", incidentno)
If oIncident.AppendToRepeatedNode(strNodeName, strError) Then
   MessageBox.Show("Node appended successfully")
End If
oIncident.SaveVariables()
```

DeleteIncident

Method: DeleteIncident	

Method description: This method deletes the loaded incident from Ultimus BPM Database if the incident has been completed or aborted.
C# method call:
`public bool DeleteIncident(out string strError)`
VB.NET method call:
`Public Function DeleteIncident(ByRef strError As String) As Boolean`	

Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to delete the incident.
Return Conditions	
`DeleteIncident` returns `TRUE` if the incident could be deleted. `DeleteIncident` returns `FALSE` if the incident could not be deleted.	

DirectActivateStep

Method: DirectActiveStep	

Method description: This method directly activates a step in a process.
C# method call:
`public bool DirectActivateStep (string strStepLabel)`
VB.NET method call:
`Public Function DirectActivateStep(ByVal strStepLabel As String) As Boolean`	

Input	
`strStepLabel`	**Input Type:** System.String **Description:** `strStepLabel` is a string that uniquely identifies a step for a workflow process.
Return Conditions	
`DirectActivateStep` returns `TRUE` if `DirectActivateStep` successfully activates the specified step. `DirectActivateStep` returns `FALSE` if `DirectActivateStep` is not able to activate the specified step.	
Comments	
`DirectActivateStep` is not to be used for Junction or Maplet steps.	

DirectActivateStepEx

Method: DirectActivateStepEx	

Method description: This method directly activates a step in a process. This method is a replacement of DirectActivateStep (for EIK Context only).
 	
C# method call:
`public bool DirectActivateStepEx (Ref Ultimus.WFServer.Task objPrevTask, string strStepLabel)`
 	
VB.NET method call:
`Public Function DirectActivateStepEx(ByRef objPrevTask As Ultimus.WFServer.Task, ByVal strStepLabel As String) As Boolean`	

Input	
`objPrevTask`	**Input Type:** Ultimus.WFServer.Task **Description:** `objPrevTask` is the object of the task's Context.
`strStepLabel`	**Input Type:** System.String **Description:** `strStepLabel` is a string that uniquely identifies a step for a workflow process.
Return Conditions	
`DirectActivateStepEx` returns `TRUE` if `DirectActivateStepEx` is able to Activate the step successfully. `DirectActivateStepEx` returns `FALSE` if `DirectActivateStepEx` is unable to activate the step successfully.	
Comments	
`DirectActivateStepEx` is not to be used for Junction or Maplet steps.	

GetIncidentStatus

Method: GetIncidentStatus	

Method description: This method returns the current status of the incident.
 	
C# method call:
`public bool GetIncidentStatus(out Ultimus.WFServer.Incident.Status status1)`
 	
VB.NET method call:
`Public Function GetIncidentStatus(ByRef status1 As Ultimus.WFServer.Incident.Status) As Boolean`	

Output	
`status1`	**Output Type:** instance of *`Incident.Status` class* (outlined on page 30) **Description:** `status1` returns the current status of the incident. The status parameter on return contains the IncidentStatus object.
Return Conditions	
`GetIncidentStatus` returns `TRUE` if `GetIncidentStatus` was able to execute and the specified incident exists in Ultimus BPM Database.	

Method: GetIncidentStatus

Example

```
'This code snippet is written in VB.NET
Dim oIncident As New Ultimus.WFServer.Incident
Dim oStatus As New Ultimus.WFServer.Incident.Status
Dim oStepType As Ultimus.WFServer.StepTypes
Dim oStepStatus As Ultimus.WFServer.Incident.Status.StepStatus
Dim i As Integer
Dim AllSteps(10) As String
Dim StepHist As String

oIncident.LoadIncident("processname", incidentno)
If oIncident.GetIncidentStatus(oStatus) Then
   For i = 0 To oStatus.TaskStatuses.Length - 1
      oStepStatus = oStatus.TaskStatuses(i)
      If oStepStatus.nStepType = oStepType.STEP_TYPE_BEGIN Then
         StepHist = oStepStatus.strStepName & " is a Begin step"
      ElseIf oStepStatus.nStepType = oStepType.STEP_TYPE_USER Then
         StepHist = oStepStatus.strStepName & " is a User Step"
      ElseIf oStepStatus.nStepType = oStepType.STEP_TYPE_PROCESS Then
         StepHist = oStepStatus.strStepName & " is a Process Step"
      ElseIf oStepStatus.nStepType = oStepType.STEP_TYPE_JUNCTION Then
         StepHist = oStepStatus.strStepName & " is a Junction Step"
      ElseIf oStepStatus.nStepType = oStepType.STEP_TYPE_FLOBOT Then
         StepHist = oStepStatus.strStepName & " is a Flobot Step"
      ElseIf oStepStatus.nStepType = oStepType.STEP_TYPE_END Then
         StepHist = oStepStatus.strStepName & " is an End Step"
      End If
      MessageBox.Show(StepHist)
   Next
End If
```

GetIncidentXML

Method: GetIncidentXML	

Method description: This method retrieves the incident XML.
 	
C# method call:
`public bool GetIncidentXML(out string strXML, out string strError)`
 	
VB.NET method call:
`Public Function GetIncidentXML(ByRef strXML As String, ByRef strError As String) As Boolean`	

Output	
`strXML`	**Output Type:** System.String **Description:** `strXML` returns the incident XML.
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to return the incident XML.
Return Conditions	
`GetIncidentXML` returns `TRUE` if the incident XML is returned successfully. `GetIncidentXML` returns `FALSE` if the incident XML does not return successfully.	
Example	

```
'This code snippet is written in VB.NET
Dim oIncident As New Ultimus.WFServer.Incident
Dim strXML As String = ""
Dim strError As String = ""

oIncident.LoadIncident("processname", incidentno)
If oIncident.GetIncidentXML(strXML, strError) Then
   MessageBox.Show(strXML)
End If
```

GetMapVersion

Method: GetMapVersion

Method description: This method retrieves the version of the process map.
C# method call:
`public uint GetMapVersion()`
VB.NET method call:
`Public Function GetMapVersion() As UInteger`

Return Conditions
`GetMapVersion` returns the map version of the loaded incident.
Example

`'This code snippet is written in VB.NET`
`Dim oIncident As New Ultimus.WFServer.Incident`
`Dim nVersion As Integer`
`oIncident.LoadIncident("processname", incidentno)`
`nVersion = oIncident.GetMapVersion()`
`MessageBox.Show(nVersion)`

GetMemo

Method: GetMemo	

Method description: This method retrieves a specified incident memo.
C# method call:
`public bool GetMemo(out string strMemo, out string strError)`
VB.NET method call:
`Public Function GetMemo(ByRef strMemo As String, ByRef strError As String) As Boolean`	

Output	
`strMemo`	**Output Type:** System.String **Description:** `strMemo` returns the incident memo.
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to retrieve the incident memo.

Method: GetMemo
Return Conditions
`GetMemo` returns `TRUE` if the incident memo is retrieved successfully. `GetMemo` returns `FALSE` if the incident memo is not retrieved successfully.
Example

```
'This code snippet is written in VB.NET
Dim oIncident As New Ultimus.WFServer.Incident
Dim strMemo As String = ""
Dim strError As String = ""

oIncident.LoadIncident("processname", incidentno)
If oIncident.GetMemo(strMemo, strError) Then
   MessageBox.Show(strMemo)
End If
```

GetNodeValue

Method: GetNodeValue	

Method description: This method retrieves the value of a specified Incident data node.
C# method call:
`public bool GetNodeValue(string strNodeName, out object objRetValue, out string strError)`
VB.NET method call:
`Public Function GetNodeValue(ByVal strNodeName As String, ByRef objRetValue As Object, ByRef strError As String) As Boolean`	

Input	
`strNodeName`	**Input Type:** System.String **Description:** `strNodeName` represents the name of the Incident data node whose value is to be retrieved.
Output	
`objRetValue`	**Output Type:** System.Object **Description:** `objRetValue` represents the value of the specified node.
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to get the value of the node.

Method: GetNodeValue
Return Conditions
`GetNodeValue` returns `TRUE` if the value for the Incident data node is returned successfully. `GetNodeValue` returns `FALSE` if the value for the Incident data node is not returned successfully.
Example

```
'This code snippet is written in VB.NET
Dim oIncident As New Ultimus.WFServer.Incident
Dim strNodeName As String = "IncidentData.Global.NodeName"
Dim retValue As Object = Nothing
Dim strError As String = ""

oIncident.LoadIncident("processname", incidentno)
If oIncident.GetNodeValue(strNodeName, retValue, strError) Then
   MessageBox.Show(retValue.ToString)
End If
```

GetProcessSchema

Method: GetProcessSchema

Method description: This method retrieves a specified incident's process XML schema.
 	
C# method call:
`public bool GetProcessSchema(out string strSchema, out Ultimus.WFServer.UltEIKXMLResolver resolver, out string strErr)`
 	
VB.NET method call:
`Public Function GetProcessSchema(ByRef strSchema As String, ByRef resolver As Ultimus.WFServer.UltEIKXMLResolver, ByRef strErr As String) As Boolean`

Output	
`strSchema`	**Output Type:** System.String **Description:** `strSchema` represents the process XML schema.
`Resolver`	**Output Type:** Ultimus.WFServer.UltEIKXMLResolver **Description:** `Resolver` retrieves an object of the `UltEIKXMLResolver` class.
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to retrieve the process XML schema.

Method: GetProcessSchema
Return Conditions
`GetProcessSchema` returns `TRUE` if the process XML schema is returned successfully. `GetProcessSchema` returns `FALSE` if the process XML schema is not returned successfully.
Example

```
'This code snippet is written in VB.NET
Dim oIncident As New Ultimus.WFServer.Incident
Dim strSchema As String = ""
Dim resolver As UltEIKXMLResolver = Nothing
Dim strError As String = ""

oIncident.LoadIncident("processname", incidentno)
If oIncident.GetProcessSchema(strSchema, resolver, strError) Then
   MessageBox.Show(strSchema)
End If
```

GetRepeatedNodeSize

Method: GetRepeatedNodeSize	

Method description: This method returns the size of a repeated Incident node.
C# method call:
`public long GetRepeatedNodeSize(string strRepeatedNodePath, out string strError)`
VB.NET method call:
`Public Function GetRepeatedNodeSize(ByVal strRepeatedNodePath As String, ByRef strError As String) As Long`	

Input	
`strRepeatedNodePath`	**Input Type:** System.String **Description:** `strRepeatedNodePath` represents the Incident path of a repeated node whose size is to be determined.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to get the size of the repeated node.

Method: GetRepeatedNodeSize
Return Conditions
`GetRepeatedNodeSize` returns the size of the specified Incident level repeated node.
Example

```
'This code snippet is written in VB.NET
Dim oIncident As New Ultimus.WFServer.Incident
Dim strRepeatedNodePath As String ="IncidentData.Global.RepeatedNodeName"
Dim strError As String = ""
Dim lNodeSize As Integer

oIncident.LoadIncident("processname", incidentno)
lNodeSize = oIncident.GetRepeatedNodeSize(strRepeatedNodePath, strError)
MessageBox.Show(lNodeSize)
```

GetStepStatus

Method: GetStepStatus	

Method description: This method retrieves the status of a specified step as an integer value.
C# method call:
`public bool GetStepStatus(string strStepLabel, string strStepID, out int nStepStatus)`
VB.NET method call:
`Public Function GetStepStatus(ByVal strStepLabel As String, ByVal strStepID As String, ByRef nStepStatus As Integer) As Boolean`	

Input	
`strStepLabel`	**Input Type:** System.String **Description:** `strStepLabel` represents the name of the step whose status is required.
`strStepID`	**Input Type:** System.String **Description:** `strStepID` represents the ID of the step whose status is required.

Method: GetStepStatus	
Output	
`nStepStatus`	**Output Type:** System.Int32 **Description:** `nStepStatus` returns an integer value which can be mapped with ENUMs in class `Ultimus.WFServer.TaskSubStatuses` to represent step status.
Return Conditions	
`GetStepStatus` returns `TRUE` if the step status could be retrieved successfully. `GetStepStatus` returns `FALSE` if the step status could not be retrieved successfully.	
Example	

```
'This code snippet is written in VB.NET
Dim oIncident As New Ultimus.WFServer.Incident
Dim incidentno As Integer
Dim nStepStatus As Integer
nStepStatus = -1
oIncident.LoadIncident("processname", incidentno)
If oIncident.GetStepStatus("StepLabel", "StepID", nStepStatus) Then
   MessageBox.Show("Step Status retrieved Sucessfully " + nStepStatus)
```

InsertIntoRepeatedNode

Method: InsertIntoRepeatedNode	

Method description: This method inserts a node at a specified index of a repeated XML schema element.
C# method call:
`public bool InsertIntoRepeatedNode(string strRepeatedNodePath, long lIndex, out string strError)`
VB.NET method call:
`Public Function GetRepeatedNodeSize(ByVal strRepeatedNodePath As String, ByRef strError As String) As Long`	

Input	
`strRepeatedNodePath`	**Input Type:** System.String **Description:** `strRepeatedNodePath` represents the complete path of the repeated node.
`lIndex`	**Input Type:** System.Int64 **Description:** `lIndex` represents where the node is to be added.

Method: InsertIntoRepeatedNode	
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to insert into a repeated node.
Return Conditions	
`InsertIntoRepeatedNode` returns `TRUE` if the node is added successfully. `InsertIntoRepeatedNode` returns `FALSE` if the node is not added successfully.	
Example	

```
'This code snippet is written in VB.NET
Dim oIncident As New Ultimus.WFServer.Incident
Dim strNodePath As String = "IncidentData.Global.nodename"
Dim strError As String = ""

oIncident.LoadIncident("processname", incidentno)

If oIncident.InsertIntoRepeatedNode(strNodePath, nodeIndex, strError) Then
   MessageBox.Show("Node inserted successfully")
End If
```

LoadIncident

Method: LoadIncident	

Method description: This method loads a specific incident of a specified process within this object.
 	
C# method call:
`public bool LoadIncident(string strProcessName, int nIncidentNo)`
 	
VB.NET method call:
`Public Function LoadIncident(ByVal strProcessName As String, ByVal nIncidentNo As Integer) As Boolean`	

Input	
`strProcessName`	**Input Type:** System.String **Description:** `strProcessName` is the name of the process whose incident is to be loaded.
`nIncidentNo`	**Input Type:** System.Int32 **Description:** `nIncidentNo` is the incident number which needs to be loaded.
Return Conditions	
`LoadIncident` returns `TRUE` if `LoadIncident` was able to execute and the specified incident exists in Ultimus BPM Database. `LoadIncident` returns `FALSE` if `LoadIncident` could not execute or the specified incident does not exist in Ultimus BPM Database.	

RemoveRepeatedNodeIndex

Method: RemoveRepeatedNodeIndex	

Method description: This method removes the node from a specified index of a repeated element.
 	
C# method call:
`public bool RemoveRepeatedNodeIndex(string strRepeatedNodePath, long lIndex, out string strError)`
 	
VB.NET method call:
`Public Function RemoveRepeatedNodeIndex(ByVal strRepeatedNodePath As String, ByVal lIndex As Long, ByRef strError As String) As Boolean`	

Input	
`strRepeatedNodePath`	**Input Type:** System.String **Description:** `strRepeatedNodePath` represents the complete path of the repeated node.
`lIndex`	**Input Type:** System.Int64 **Description:** `lIndex` represents the index number from which node is to be removed.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to remove the node from the specified index.
Return Conditions	
`RemoveRepeatedNodeIndex` returns `TRUE` if the node at the specified index is removed successfully. `RemoveRepeatedNodeIndex` returns `FALSE` if the node at the specified index is not removed successfully.	
Example	

```
'This code snippet is written in VB.NET
Dim oIncident As New Ultimus.WFServer.Incident
Dim strNodePath As String = "IncidentData.Global.NodeName"
Dim strError As String = ""

oIncident.LoadIncident("processname", incidentno)
If oIncident.RemoveRepeatedNodeIndex(strNodePath, nodeIndex, strError) Then
   MessageBox.Show("Node removed successfully")
End If
```

SaveVariables

Method: SaveVariables

Method description: This method saves the changes made to the incident data without the need for the EIK user to submit the task. Note that this method should be called to permanently commit the changes after updating any XML schema element in the Incident data.
 	
C# method call:
`public bool SaveVariables()`
 	
VB.NET method call:
`Public Function SaveVariables() As Boolean`

Return Conditions
`SaveVariables` returns `TRUE` if the changes made to incident data are committed successfully. `SaveVariables` returns `FALSE` if the changes made to incident data are not committed successfully.

SetIncidentXML

Method: SetIncidentXML	

Method description: This method sets a specified incident's XML data.
 	
C# method call:
`public bool SetIncidentXML(string strXML, bool bValidateXML, out string strError)`
 	
VB.NET method call:
`Public Function SetIncidentXML(ByVal strXML As String, ByVal bValidateXML As Boolean, ByRef strError As String) As Boolean`	

Input	
`strXML`	**Input Type:** System.String **Description:** `strXML` represents the updated XML.
`bValidateXML`	**Input Type:** System.Int64 **Description:** `bValidateXML` represents a Boolean value indicating whether to validate the given XML or not. If `TRUE`, the XML is validated; if `FALSE`, the XML is not validated.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to set the Incident XML.

Method: SetIncidentXML
Return Conditions
`SetIncidentXML` returns `TRUE` if the incident XML is updated successfully. `SetIncidentXML` returns `FALSE` if the Incident XML is not updated successfully.
Example

```
'This code snippet is written in VB.NET
Dim oIncident As New Ultimus.WFServer.Incident
Dim strError As String = ""
Dim strXML As String = ""

oIncident.LoadIncident("processname", incidentno)
oIncident.GetIncidentXML(strXML, strError)
'strXML containing the updated incident XML Data
If oIncident.SetIncidentXML(strXML, False, strError) Then
   MessageBox.Show("Data updated successfully")
End If
```

SetNodeValue

Method: SetNodeValue	

Method description: This method sets a specified node's value.
C# method call:
`public bool SetNodeValue(string strNodeName, object objValue, out string strError)`
VB.NET method call:
`Public Function SetNodeValue(ByVal strNodeName As String, ByVal objValue As Object, ByRef strError As String) As Boolean`	

Input	
`strNodeName`	**Input Type:** System.String **Description:** `strNodeName` represents the name of the node whose value is to be updated.
`objValue`	**Input Type:** System.Object **Description:** `objValue` represents the value with which the specified node is to be updated.

Method: SetNodeValue	
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to set the specified node value.
Return Conditions	
`SetNodeValue` returns `TRUE` if the value is updated successfully. `SetNodeValue` returns `FALSE` if the value is not updated successfully.	
Example	

```
'This code snippet is written in VB.NET
Dim oIncident As New Ultimus.WFServer.Incident
Dim strNodeName As String = "IncidentData.Global.NodeName"
Dim objValue As Object = "Value set through code"
Dim strError As String = ""

oIncident.LoadIncident("processname", incidentno)
If oIncident.SetNodeValue(strNodeName, objValue, strError) Then
   MessageBox.Show("node data updated successfully")
End If
```

Properties and method in the `Incident.Status` class

The `Incident.Status` class describes the status of an incident of a business process. It is returned with the call to `GetIncidentStatus` method of `WFServer.Incident` class. Outlined below are the properties and method for the `Incident.Status` class (in alphabetical order):

- *Properties for the `Incident.Status` class*
- *GetGraphicalStatus*

Properties for the `Incident.Status` class

Table 10. Properties for the `Incident.Status` class

Property Name	Property Type and Description
`dCompleteTime`	**Type:** double (date/time stamp) **Description:** `dCompleteTime` corresponds with the time when the incident was completed.
`dLateTime`	**Type:** double (date/time stamp) **Description:** `dLateTime` corresponds with the time after which the incident becomes urgent.
`dStartTime`	**Type:** double (date/time stamp) **Description:** `dStartTime` corresponds with the time the incident was initiated.

Table 10. Properties for the `Incident.Status` class *(Continued)*

Property Name	Property Type and Description
`nIncidentStatus`	**Type:** System.Int32 **Description:** `nIncidentStatus` corresponds to the status of the incident. Returned values correspond to values in the `Ultimus.WFServer.IncidentStatuses` *Enumerations (ENUMs)* (outlined on page 27). For this property value (and also in Ultimus BPM Database), the status of an incident is represented as an integer value. The status value of an incident is bitwise (meaning, the numerical incident status value is a summation of the incident's various states). For example, if an active incident (`status` = 1) is also late (`status` = 16), then the represented incident status value will be 17. Examples of numerical incident status values are: • `Active` = 1 • `Completed` = 3 • `Late` = 16 • `Stalled` = 32

`TaskStatuses`	**Type:** instance of the *Properties in the `Incident.Status.StepStatus` class* (outlined on page 57)
	
	Description: `TaskStatuses` is an array which contains the status information for each step in the incident.
	
	Example:
	`'This code snippet is written in VB.NET`
	`Dim FindIncident As Ultimus.WFServer.Incident`
	`Dim RetIncStatus As Ultimus.WFServer.Incident.Status`
	`Dim S As Ultimus.WFServer.Incident.Status.StepStatus`
	`Dim NumSteps as Integer`
	`Dim StepLabel as String`
	`Dim xx as Integer`
	`FindIncident = New Ultimus.WFServer.Incident`
	`FindIncident.LoadIncident("myProcessName",2)`
	`FindIncident.GetIncidentStatus(RetIncStatus)`
	`NumSteps = RetIncStatus.TaskStatuses.Length`
	`For xx = 0 To NumSteps - 1`
	`S = RetIncStatus.TaskStatuses(xx)`
	`StepLabel = S.strStepName`
	`Next xx`

GetGraphicalStatus

Method: GetGraphicalStatus	

Method description: This method generates the graphical status of the incident.
 	
C# method call:
`public bool GetGraphicalStatus(string strProcessName, int nIncidentNo, int nVersion, out byte[] bytesGif)`
 	
VB.NET method call:
`Public Function GetGraphicalStatus(ByVal strProcessName As String, ByVal nIncidentNo As Integer, ByVal nVersion As Integer, ByRef bytesGif As Byte( )) As Boolean`	

Input	
`strProcessname`	**Input Type:** System.String **Description:** `strProcessname` is the name of the Ultimus process to which a graphical status is to be retrieved.
`nIncidentNo`	**Input Type:** System.Int32 **Description:** `nIncidentNo` is the incident number of the Ultimus process to which a graphical status is to be retrieved.
`nVersion`	**Input Type:** System.Int32 **Description:** `nVersion` is the version number of the Ultimus process to which a graphical status is to be retrieved.
Output	
`bytesGIF`	**Output Type:** byte **Description:** `bytesGIF` on return contains an array of bytes containing a `GIF` file representing the incident status.
Return Conditions	
`GetGraphicalStatus` returns `TRUE` if `GetGraphicalStatus` was able to execute. `GetGraphicalStatus` returns `FALSE` if `GetGraphicalStatus` could not execute.	

Method: GetGraphicalStatus
Example

```
'This code snippet is written in VB.NET
Dim RetIncidentStatus As New Ultimus.WFServer.Incident.Status
Dim GraphIncStatus As Byte()
RetIncidentStatus.GetGraphicalStatus("MyProcess",1,1, GraphIncStatus)
'Graph retrieved, now save it to a file
Dim FileName As String
FileName = "f:\MyOutput.gif"
Dim fs = New FileStream(FileName, FileMode.Create, FileAccess.Write)
fs.Write(GraphIncStatus, 0, GraphIncStatus.Length)
fs.close()
```

Properties in the `Incident.Status.StepStatus` class

The `Incident.Status.StepStatus` class describes the status of a step within an Ultimus process incident. Outlined below are the properties for the `Incident.Status.StepStatus` class (in alphabetical order).

Table 11. Properties for the `Incident.Status.StepStatus` class

Property Name	Property Type and Description
`dEndTime`	**Type:** double (date/time stamp) **Description:** `dEndTime` corresponds to the time when the step within the Ultimus process incident was completed.
`dStartTime`	**Type:** double (date/time stamp) **Description:** `dStartTime` corresponds to the time when the step was activiated.

Table 11. Properties for the `Incident.Status.StepStatus` class *(Continued)*

Property Name	Property Type and Description

`nStepProperties`	**Type:** System.Int32
	
	Description: `nStepProperties` pertains to the properties of a step within an Ultimus process incident. Returned values correspond to values in the *Enumerations for `WFServer.StepProps` class* (outlined on page 28).
	
	Example:
	`'This code snippet is written in VB.NET`
	`'In case the step has multiple true properties,`
	`'construct the Enum comparison in the following manner`
	`If ((AllStepStatuses.nStepProperties And Ultimus.WFServer.StepProps.STEP_PROPERTY_PRIVATE) = Ultimus.WFServer.StepProps.STEP_PROPERTY_PRIVATE) Then`
	`MessageBox.Show("Task is private")`
	`End If`

`nStepStatus`	**Type:** System.Int32
	
	Description: `nStepStatus` pertains to the current status of a step within an Ultimus process incident. Returned values correspond to values in the *Enumerations for `WFServer.TaskStatuses` class* (outlined on page 29).
	
	Example:
	`If AllStepStatuses.nStepStatus = TaskStatEnum.TASK_STATUS_COMPLETED Then`
	`MyLabel.Text = "Completed"`
	`End If`

`nStepSubStatus`	**Type:** System.Int32
	
	Description: `nStepStatus` pertains to the current status of a step within an Ultimus process incident. Returned values correspond to values in the *Enumerations for `WFServer.TaskStatuses` class* (outlined on page 29).
	
	Example:
	`If ( (AllStepStatuses.nStepSubStatus And TaskSubStatEnum.TASK_STATUS_COMPLETE_INCIDENT_ACTIVE) = TaskSubStatEnum.TASK_STATUS_COMPLETE_INCIDENT_ACTIVE) Then`
	`Label28.Text = "Step Complete, Incident still Active"`
	`End If`

Table 11. Properties for the `Incident.Status.StepStatus` class *(Continued)*

Property Name	Property Type and Description

`nStepType`	**Type:** System.Int32
	
	Description: `nStepType` pertains to the type of a step within an Ultimus process incident. Returned values correspond to values in the *Enumerations for `WFServer.StepTypes` class* (outlined on page 28).
	
	Example:
	`If AllStepStatuses.nStepType = StepTypesEnum.STEP_TYPE_USER Then`
	`  MyLabel.Text = "User Step"`
	`End If`

`strStepName`	**Type:** System.String **Description:** `strStepName` pertains to the name of the step within the Ultimus process.
`strStepRecipient`	**Type:** System.String **Description:** `strStepRecipient` pertains to the name of the user who this step in the Ultimus process incident was activated or who completed the step.

Properties and methods in the `Task` class

The `Task` class represents a task of an Ultimus process incident. All interactions with a task occur through this class. Outlined below are the properties for the `Task` class (in alphabetical order):

- *Properties for the `Task` class*
- *AbortStep*
- *AddMemoEntry*
- *AppendToRepeatedNode*
- *AssignQueueTask*
- *AssignTask*
- *CheckInTask*
- *CheckOutTask*
- *ConferTask*
- *DeleteTask*
- *ExtractFormURL*
- *GetCheckedOutUser*

- *GetMapVersion*
- *GetMemo*
- *GetNodeValue*
- *GetProcessHelpURL*
- *GetRepeatedNodeSize*
- *GetStepHelpURL*
- *GetStepSchema*
- *GetTaskXML*
- *InitializeFromInitiateTaskId*
- *InitializeFromNextQueueTask*
- *InitializeFromTaskId*
- *InsertIntoRepeatedNode*
- *PutTaskInQueue*
- *Refresh*
- *RemoveRepeatedNodeIndex*
- *Return*
- *SaveTemplate*
- *SaveXML*
- *Send*
- *SendFrom*
- *SetNodeValue*
- *SetTaskXML*
- *TakeBack*

Properties for the `Task` class

Table 12. Properties for the `Task` class

Property Name	Property Type and Description
`dCost`	**Type:** double **Description:** `dCost` is the calculated cost of performing the task, corresponding to the entered amount of task time by the Ultimus form user.
`dDelayTime`	**Type:** double (date/time stamp) **Description:** dDelayTime corresponds to the time before the task status changes to Active after the completion of the previous step.
`dEndTime`	**Type:** double (date/time stamp) **Description:** `dEndTime` corresponds to the time when the step within the Ultimus process incident was completed.
`dOverDueTime`	**Type:** double (date/time stamp) **Description:** `dOverDueTime` corresponds to the time when the step becomes overdue.
`dQueueEndTime`	**Type:** double (date/time stamp) **Description:** `dQueueEndTime` corresponds to the time when a user takes the task from the queue (the recipient changes from selected Queue Group to the user who picked the task).
`dQueueStartTime`	**Type:** double (date/time stamp) **Description:** `dQueueStartTime` corresponds to the time when the task was put in a queue.
`dStartTime`	**Type:** double (date/time stamp) **Description:** `dStartTime` pertains to the time when the task was activated.
`dUrgentTime`	**Type:** double (date/time stamp) **Description:** `dUrgentTime` corresponds to the time when the step completion time expires.
`nIncidentNo`	**Type:** System.Int32 **Description:** `nIncidentNo` pertains to the incident number of the Ultimus process to which the task is associated.
`nPriority`	**Type:** System.Int32 **Description:** `nPriority` pertains to the current priority of the incident to which the task is associated.

Table 12. Properties for the `Task` class *(Continued)*

Property Name	Property Type and Description
`nProcessVersion`	**Type:** System.Int32 **Description:** `nProcessVersion` corresponds to the version of the Ultimus process to which this task is associated.
`nRecipientType`	**Type:** System.Int32 **Description:** `nRecipientType` corresponds to an integer value denoting the type of recipient for the task. Possible values include: **Type of recipient** — **Value** User — 0 Recipient of step — 0 Recipient of previous step — 0 Dynamic recipient — 0 Supervisor of dynamic recipient — 0 Manager of dynamic recipient — 0 Job function — 1 Supervisor of step — 1 Manager of step — 1 Supervisor of previous step — 1 Manager of previous step — 1 Group — 2 Queue — 3 Weighted round robin — 10 First available person — 11 Department — 14 First available online person — 23

`nStepProperties`	**Type:** System.Int32
	
	Description: `nStepProperties` corresponds to the properties of a step to which this task is associated. Returned values correspond to values in the *Enumerations for `WFServer.StepProps` class* (outlined on page 28).
	
	Example:
	`'This code snippet is written in VB.NET.`
	`If`
	`(`
	`MyTask.nStepProperties And`
	`StepPropsEnum.STEP_PROPERTY_PRIVATE) =`
	`StepPropsEnum.STEP_PROPERTY_PRIVATE`
	`)`
	`Then`
	`MyLabel.Text = "This is a private step"`
	`End If`

Table 12. Properties for the `Task` class *(Continued)*

Property Name	Property Type and Description
`nStepType`	**Type:** System.Int32 **Description:** `nStepType` pertains to the type of the step to which this task is associated. Returned values correspond to the values in the *Enumerations for `WFServer.StepTypes` class* (outlined on page 28).

`nTaskStatus`	**Type:** System.Int32
	
	Description: `nTaskStatus` pertains to the current status of a task within an Ultimus process incident. Returned values correspond to values in the *Enumerations for `WFServer.TaskStatuses` class* (outlined on page 29).
	
	Example:
	`'This code snippet is written in VB.NET.`
	`If MyTask.nTaskStatus = TaskStatEnum.TASK_STATUS_COMPLETED Then`
	`  MyLabel.Text = "Task is Completed"`
	`End If`

`nTaskRate`	**Type:** System.Int32 **Description:** `nTaskRate` pertains to the cost of the task and per unit of time as defined by the designer of the Ultimus process.

Table 12. Properties for the `Task` class *(Continued)*

Property Name	Property Type and Description

`nTaskSubStatus`	**Type:** System.Int32
	
	Description: `nTaskSubStatus` pertains to the current substatus of a task within an Ultimus process incident. Returned values correspond to values in the *Enumerations for `WFServer.TaskSubStatuses` class* (outlined on page 30).
	
	Example:
	`'This code snippet is written in VB.NET.`
	`Dim TaskSubStat as Ultimus.WFServer.TaskSubStatus`
	`If`
	`(`
	`(MyTask.nTaskSubStatus And`
	`TaskSubStat.TASK_STATUS_COMPLETE_INCIDENT_COMPLETE)`
	`= TaskSubStat.TASK_STATUS_COMPLETE_INCIDENT_COMPLETE`
	`)`
	`Then`
	`MessageBox.Show("Task Complete, and Incident Complete")`
	`Else`
	`MessageBox.Show("Task Complete, Incident still Active")`
	`End If`

`nTaskTime`	**Type:** System.Int32 **Description:** `nTaskTime` pertains to the number of minutes in which the task took to become complete (as entered by the user while completing the task).
`strAssignedToUser`	**Type:** System.String **Description:** `strAssignedToUser` contains the short name of the user assigned the task.
`strAssignedToUserFullName`	**Type:** System.String **Description:** `strAssignedToUser` contains the full name of the user assigned the task.

Table 12. Properties for the `Task` class *(Continued)*

Property Name	Property Type and Description
`strFormUrl`	**Type:** System.String **Description:** `strFormUrl` corresponds to the URL of the form associated with the task. Opening an Ultimus Form using the `strFormURL` property may prompt for the form user for user name and password authentication. The Ultimus System Administrator setting "Form Authentication Timeout" mandates how often the form user is prompted for authentication credentials. Refer to *Configuring the Advanced node properties* section in *Ultimus System Administrator Help* for more information regarding this setting.
`strGroupName`	**Type:** System.String **Description:** `strGroupName` represents the name of the group assigned to the task as recipient of that step. `strGroupName` pertains to the name of the group associated with the step.
`strHelpUrl`	**Type:** System.String **Description:** `strHelpUrl` is the step level help value corresponding to the value in the "Help URL" field of the step associated with the task.
`strProcessName`	**Type:** System.String **Description:** `strProcessName` pertains to the name of the Ultimus process to which this task is associated.
`strRecipient`	**Type:** System.String **Description:** `strRecipient` pertains to the short name of the user to whom this task was originally activated.
`strRecipientFullName`	**Type:** System.String **Description:** `strRecipientFullName` pertains to the full name of the user to whom this task was originally activated.
`strStepId`	**Type:** System.String **Description:** `strStepId` pertains to the ID of the step within an Ultimus process incident to which this task is associated.
`strStepName`	**Type:** System.String **Description:** `strStepName` pertains to the name of the step within an Ultimus process to which this task is associated.
`strStepOrg`	**Type:** System.String **Description:** `strStepOrg` pertains to the organization to which the recipient of the step belongs.

Table 12. Properties for the `Task` class *(Continued)*

Property Name	Property Type and Description
`strSummary`	**Type:** System.String **Description:** The incident summary for this incident can be set by specifying a value for `strSummary`.
`strTaskId`	**Type:** System.String **Description:** `strTaskId` pertains to the unique ID of this task.
`strUser`	**Type:** System.String **Description:** `strUser` is the name of the recipient corresponding to the value in the "Recipient" field of the step associated with the task.
`strUserFullName`	**Type:** System.String **Description:** `strUserFullName` is the full name of the recipient corresponding to the value in the "Recipient" field of the step associated with the task.

AbortStep

Method: AbortStep	

Method description: This method aborts the step associated with the task.
C# method call:
`public bool AbortStep(out string strError)`
VB.NET method call:
`Public Function AbortStep(ByRef strError As String) As Boolean`	

Output	
`strError`	**Output Type:** System.String **Description:** `strError` is a string that contains an error message if the task's step cannot be aborted.
Return Conditions	
`AbortStep` returns `TRUE` if `AbortStep` was able to complete successfully. `AbortStep` returns `FALSE` if `AbortStep` could not execute or if it could not abort the task.	

AddMemoEntry

Method: AddMemoEntry	

Method description: This method adds a memo entry for a specific step.
 	
C# method call:
`public bool AddMemoEntry(string strUser, string strStepLabel, string strMemoEntry, out string strError)`
 	
VB.NET method call:
`Public Function AddMemoEntry(ByVal strUser As String, ByVal strStepLabel As String, ByVal strMemoEntry As String, ByRef strError As String) As Boolean`	

Input	
`strUser`	**Input Type:** System.String **Description:** `strUser` represents the name of the user against which an entry is made.
`strStepLabel`	**Input Type:** System.String **Description:** `strStepLabel` represents the name of the step for which the memo is to be updated.
`strMemoEntry`	**Input Type:** System.String **Description:** `strMemoEntry` represents the text to be added as the memo entry.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to add a memo.
Return Conditions	
`AddMemoEntry` returns `TRUE` if an entry is made successfully. `AddMemoEntry` returns `FALSE` if an entry is not made successfully.	

Method: AddMemoEntry
Example

```
'This code snippet is written in VB.NET
Dim oTask As New Ultimus.WFServer.Task
Dim strTaskId As String
Dim strError As String = ""

If oTask.InitializeFromTaskId(strTaskId) Then
   If oTask.AddMemoEntry("UserName", "StepName", "MemoDetails", strError) Then
   MessageBox.Show("Memo has been added successfully")
   End If
   'commit the changes to the task data
   oTask.Refresh(strError)
End If
```

AppendToRepeatedNode

Method: AppendToRepeatedNode	

Method description: This method appends a node to the specified repeated task element.
C# method call:
`public bool AppendToRepeatedNode(string strRepeatedNodePath, out string strError)`
VB.NET method call:
`Public Function AppendToRepeatedNode(ByVal strRepeatedNodePath As String, ByRef strError As String) As Boolean`	

Input	
`strRepeatedNodePath`	**Input Type:** System.String **Description:** `strRepeatedNodePath` represents the path of the repeated node where an additional node is to be appended.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to append to a repeated node.

Method: AppendToRepeatedNode
Return Conditions
`AppendToRepeatedNode` returns `TRUE` if the node is appended successfully. `AppendToRepeatedNode` returns `FALSE` if the node is not appended successfully.
Example

```
'This code snippet is written in VB.NET
Dim oTask As New Ultimus.WFServer.Task
Dim strTaskId As String
Dim strRepNodePath As String = "TaskData.Global.NodeName"
Dim strError As String = ""

If oTask.InitializeFromTaskId(strTaskId) Then
   If oTask.AppendToRepeatedNode(strRepNodePath, strError) Then
   MessageBox.Show("Node has been appended successfully")
   End If
   'commit the changes to the task data
   oTask.Refresh(strError)
End If
```

AssignQueueTask

Method: AssignQueueTask	

Method description: If this task is a queue task, this method assigns it to a given user.
 	
C# method call:
`public bool AssignQueueTask(string strAssignToUser, out string strAssignedUser, out string strError)`
 	
VB.NET method call:
`Public Function AssignQueueTask(ByVal strAssignToUser As String, ByRef strAssignedUser As String, ByRef strError As String) As Boolean`
 	
Overload method calls
 	
C# method call:
`public bool AssignQueueTask(string strAssignToUser, string strOrgName, out string strAssignedUser, out string strError)`
 	
VB.NET method call:
`Public Function AssignQueueTask(ByVal strAssignToUser As String, ByVal strOrgName As String, ByRef strAssignedUser As String, ByRef strError As String) As Boolean`	

Input	
`strAssignToUser`	**Input Type:** System.String **Description:** `strAssignToUser` is the user's short name to whom the task is to be assigned.
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` is the name of the organization to which the user belongs. This parameter is used if the overloaded method is called.

Method: AssignQueueTask	
Output	
`strAssignedUser`	**Output Type:** System.String **Description:** `strAssignedUser` returns the short name of the user to whom the task is actually assigned. (For example, if two users simultaneously attempt to pull a task from the queue, the task is assigned only to the first user. If the value of `strAssignToUser` differs from the short name of the person intending to pull the queue task, then the `strAssigneduser` value shows who actually has the queue task.)
`strError`	**Output Type:** System.String **Description:** `strError` is a string that contains an error message if the queue task could not be assigned to the specified user.
Return Conditions	
`AssignQueueTask` returns `TRUE` if `AssignQueueTask` was able to execute. `AssignQueueTask` returns `FALSE` if `AssignQueueTask` could not execute.	

AssignTask

Method: AssignTask

Method description: This method assigns a task to another user.
 	
C# method call:
`public bool AssignTask(string strAssignedTo)`
 	
VB.NET method call:
`Public Function AssignTask(ByVal strAssignedTo As String) As Boolean`
 	
C# overload method call:
`public bool AssignTask(string strAssignedTo, string strOrgName)`
 	
VB.NET overload method call:
`Public Function AssignTask(ByVal strAssignedTo As String, ByVal strOrgName As String) As Boolean`

Method: AssignTask	
Input	
`strAssignedTo`	**Input Type:** System.String **Description:** `strAssignedTo` is the user short name to whom the task is to be assigned.
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` is the name of the organization to which the user belongs. This parameter is used if the overloaded method is called.
Return Conditions	
`AssignTask` returns `TRUE` if `AssignTask` was able to execute. `AssignTask` returns `FALSE` if `AssignTask` could not execute.	

CheckInTask

Method: CheckInTask

Method description: This method checks in a task from a specified user.

C# method call:

```
public bool CheckInTask(string strCheckedOutUser, out string strError)
```

VB.NET method call:

```
Public Function CheckInTask (ByVal strCheckedOutUser As String, ByRef strError As
String) As Boolean
```

Overload method calls

C# method call:

```
public bool CheckInTask(string strCheckedOutUser, string strXML, out string
strError)
```

VB.NET method call:

```
Public Function CheckInTask (ByVal strCheckedOutUser As String, ByVal strXML As
String, ByRef strError As String) As Boolean
```

Method: CheckInTask	
Input	
`strCheckedOutUser`	**Input Type:** System.String **Description:** `strCheckedOutUser` is the user short name of the user to whom the task is checked out.
`strXML`	**Input Type:** System.String **Description:** `strXML` specifies the task data in `XML` format.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` is a string that contains an error message if the task could not be checked in.
Comments	
When the non-overloaded `CheckInTask` method executes, current task XML data is checked in.	
Return Conditions	
`CheckInTask` returns `TRUE` if `CheckInTask` was able to execute. `CheckInTask` returns `FALSE` if `CheckInTask` could not execute.	

CheckOutTask

Method: CheckOutTask	

Method description: This method checks out a task to a specified user.
C# method call:
`public bool CheckOutTask(string strCheckOutUser, out string strError)`
VB.NET method call:
`Public Function CheckOutTask (ByVal strCheckOutUser As String, ByRef strError As String) As Boolean`	

Input	
`strCheckOutUser`	**Input Type:** System.String **Description:** `strCheckOutUser` is the user short name of the user to whom the task is to be checked out.

Method: CheckOutTask	
Output	
`strError`	**Output Type:** System.String **Description:** `strError` is a string that contains an error message if the task could not be checked out.
Return Conditions	
`CheckOutTask` returns `TRUE` if `CheckOutTask` was able to execute. `CheckOutTask` returns `FALSE` if `CheckOutTask` could not execute.	

ConferTask

Method: ConferTask	

Method description: This method confers a task to another user.
C# method call:
`public bool ConferTask(string strConferredTo, out string strError)`
VB.NET method call:
`Public Function ConferTask (ByVal strConferredTo As String, ByRef strError As String) As Boolean`	

Input	
`strConferredTo`	**Input Type:** System.String **Description:** `strConferredTo` is the user short name to whom the task is to be conferred.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` is a string that contains an error message if the task could not be conferred.
Return Conditions	
`ConferTask` returns `TRUE` if `ConferTask` was able to execute. `ConferTask` returns `FALSE` if `ConferTask` could not execute.	

DeleteTask

Method: DeleteTask

Method description: This method deletes the task from Ultimus BPM Database if the task is completed or aborted. (It is recommended that the incident should be completed or aborted before calling this method.)
 	
C# method call:
`public bool DeleteTask()`
 	
VB.NET method call:
`Public Function DeleteTask() As Boolean`

Return Conditions
`DeleteTask` returns `TRUE` if `DeleteTask` was able to execute. `DeleteTask` returns `FALSE` if `DeleteTask` could not execute or if it could not delete the task.

ExtractFormURL

Method: ExtractFormURL	

Method description: This method extracts the URL of the default form type chosen, as specified in the step properties for the task.
 	
C# method call:
`public bool ExtractFormURL(out string strURL)`
 	
VB.NET method call:
`Public Function ExtractFormURL(ByRef strURL As String) As Boolean`	

Output	
`strURL`	**Output Type:** System.String **Description:** `strURL` is the URL for the default form.
Return Conditions	
`ExtractFormURL` returns `TRUE` if `ExtractFormURL` was able to execute. `ExtractFormURL` returns `FALSE` if `ExtractFormURL` could not execute.	

GetCheckedOutUser

Method: GetCheckedOutUser	

Method description: This method aquires the short name of the user to whom a specified task is checked out.
 	
C# method call:
`public bool GetCheckedOutUser(out string strCheckedOutUser, out string strError)`
 	
VB.NET method call:
`Public Function GetCheckedOutUser (ByVal strCheckedOutUser As String, ByRef strError As String) As Boolean`	

Output	
`strCheckedOutUser`	**Output Type:** System.String **Description:** `strCheckedOutUser` is the user short name of the user to whom the task is checked out.
`strError`	**Output Type:** System.String **Description:** `strError` is a string that contains an error message if the `GetCheckdOutUser` could not execute.
Return Conditions	
`GetCheckedOutUser` returns `TRUE` if `GetCheckedOutUser` was able to execute. `GetCheckedOutUser` returns `FALSE` if the task is not currently checked out or if an error occurs.	

GetMapVersion

Method: GetMapVersion

Method description: This method retrieves the version of the process map.
 	
C# method call:
`public uint GetMapVersion()`
 	
VB.NET method call:
`Public Function GetMapVersion() As UInteger`

Return Conditions
`GetMapVersion` returns the map version of the initialized task.
Example

```
'This code snippet is written in VB.NET
Dim oTask As New Ultimus.WFServer.Task
Dim strTaskId As String
Dim strError As String = ""

If oTask.InitializeFromTaskId(strTaskId) Then
   MessageBox.Show(oTask.GetMapVersion())
End If
```

GetMemo

Method: GetMemo	

Method description: This method retrieves a specified task memo.
 	
C# method call:
`public bool GetMemo(out string strMemo, out string strError)`
 	
VB.NET method call:
`Public Function GetMemo(ByRef strMemo As String, ByRef strError As String) As Boolean`	

Output	
`strMemo`	**Output Type:** System.String **Description:** `strMemo` returns the task memo.
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to retrieve the task memo.
Return Conditions	
`GetMemo` returns `TRUE` if the task memo is retrieved successfully. `GetMemo` returns `FALSE` if the task memo is not retrieved successfully.	
Example	

```
'This code snippet is written in VB.NET
Dim oTask As New Ultimus.WFServer.Task
Dim strTaskId As String
Dim strMemo As String = ""
Dim strError As String = ""

If oTask.InitializeFromTaskId(strTaskId) Then
   If oTask.GetMemo(StrMemo, strError) Then
   MessageBox.Show(strMemo)
   End If
End If
```

GetNodeValue

Method: GetNodeValue	

Method description: This method retrieves the value of a specified Task data node.
 	
C# method call:
`public bool GetNodeValue(string strNodeName, out object objRetValue, out string strError)`
 	
VB.NET method call:
`Public Function GetNodeValue(ByVal strNodeName As String, ByRef objRetValue As Object, ByRef strError As String) As Boolean`	

Input	
`strNodeName`	**Input Type:** System.String **Description:** `strNodeName` represents the name of the Task data node whose value is to be retrieved.
Output	
`objRetValue`	**Output Type:** System.Object **Description:** `objRetValue` represents the value of the specified node.
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to get the value of the node.
Return Conditions	
`GetNodeValue` returns `TRUE` if the value for the `Task` data node is returned successfully. `GetNodeValue` returns `FALSE` if the value for the `Task` data node is not returned successfully.	

Example

```
'This code snippet is written in VB.NET
Dim oTask As New Ultimus.WFServer.Task
Dim strTaskId As String
Dim objRetValue As Object
Dim strError As String = ""

If oTask.InitializeFromTaskId(strTaskId) Then
   If oTask.GetNodeValue("NodePath", objRetValue, strError) Then
   MessageBox.Show(objRetValue)
   End If
End If
```

GetProcessHelpURL

Method: GetProcessHelpUrl	

Method description: This method returns the Ultimus process Help URL that is associated with this task.
C# method call:
`public bool GetProcessHelpURL(out string strHelpURL)`
VB.NET method call:
`Public Function GetProcessHelpURL(ByRef strHelpURL As String) As Boolean`	

Output	
`strHelpURL`	**Output Type:** System.String **Description:** `strHelpURL` is the URL for online process Help that is associated with this task.
Return Conditions	
`GetProcessHelpURL` returns `TRUE` if `GetProcessHelpURL` was able to execute. `GetProcessHelpURL` returns `FALSE` if `GetProcessHelpURL` could not execute.	

GetRepeatedNodeSize

Method: GetRepeatedNodeSize	

Method description: This method returns the size of a repeated Task node.
C# method call:
`public long GetRepeatedNodeSize(string strRepeatedNodePath, out string strError)`
VB.NET method call:
`Public Function GetRepeatedNodeSize(ByVal strRepeatedNodePath As String, ByRef strError As String) As Long`	

Input	
`strRepeatedNodePath`	**Input Type:** System.String **Description:** `strRepeatedNodePath` represents the Task path of a repeated node whose size is to be determined.

Method: GetRepeatedNodeSize	
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to get the size of the repeated node.
Return Conditions	
`GetRepeatedNodeSize` returns the size of the specified `Task` level repeated node.	

GetStepHelpURL

Method: GetStepHelpURL	

Method description: This method returns the step Help URL (if available) that is associated with this task.
C# method call:
`public bool GetStepHelpURL(out string strHelpURL)`
VB.NET method call:
`Public Function GetStepHelpURL(ByRef strHelpURL As String) As Boolean`	

Output	
`strHelpURL`	**Output Type:** System.String **Description:** `strHelpURL` is the URL for online step Help that is associated with this task.
Return Conditions	
`GetStepHelpURL` returns `TRUE` if `GetStepHelpURL` was able to execute. `GetStepHelpURL` returns `FALSE` if `GetStepHelpURL` could not execute.	

GetStepSchema

Method: GetStepSchema	

Method description: This method returns a specified step's XML schema.
 	
C# method call:
`public bool GetStepSchema(out string strSchema, out Ultimus.WFServer.UltEIKXMLResolver resolver, out string strErr)`
 	
VB.NET method call:
`Public Function GetStepSchema(ByRef strSchema As String, ByRef resolver As Ultimus.WFServer.UltEIKXMLResolver, ByRef strErr As String) As Boolean`	

Output	
`strSchema`	**Output Type:** System.String **Description:** `strSchema` represents the step XML schema.
`Resolver`	**Output Type:** Ultimus.WFServer.UltEIKXMLResolver **Description:** `Resolver` returns an object of `UltEIKXMLResolver` class.
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to get the step XML schema.
Return Conditions	
`GetStepSchema` returns `TRUE` if the step XML schema is returned successfully. `GetStepSchema` returns `FALSE` if the step XML schema is not returned successfully.	

Method: GetStepSchema
Example

```
'This code snippet is written in VB.NET
Dim oTask As New Ultimus.WFServer.Task
Dim strTaskId As String
Dim strSchema As String
Dim resolver As Ultimus.WFServer.UltEIKXMLResolver
Dim strError As String = ""

If oTask.InitializeFromTaskId(strTaskId) Then
   If oTask.GetStepSchema(strSchema, resolver, strError) Then
   MessageBox.Show(strSchema)
   End If
End If
```

GetTaskXML

Method: GetTaskXML

Method description: This method returns the XML Task data.
 	
C# method call:
`public bool GetTaskXML(out string strXML, out string strError)`
 	
VB.NET method call:
`Public Function GetTaskXML(ByRef strXML As String, ByRef strError As String) As Boolean`

Output	
`strXML`	**Output Type:** System.String **Description:** `strXML` represents the Task XML data.
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to get the Task XML data.

Return Conditions
`GetTaskXML` returns `TRUE` if the `Task` XML is returned successfully. `GetTaskXML` returns `FALSE` if the `Task` XML is not returned successfully.

InitializeFromInitiateTaskId

Method: InitializeFromInitiateTaskId	

Method description: This method initializes a task using a specified user's short user name and task ID.
 	
C# method call:
`public bool InitializeFromInitiateTaskId (string strUser, string strTaskId)`
 	
VB.NET method call:
`Public Function InitializeFromInitiateTaskId(ByVal strUser As String, ByVal strTaskId As String) As Boolean`	

Input	
`strUser`	**Input Type:** System.String **Description:** `strUser` is the user's short name to initiate the task.
Output	
`strTaskId`	**Output Type:** System.String **Description:** `strTaskId` is the task ID, which is required to initialize through this method.
Return Conditions	
`InitializeFromInitiateTaskId` returns `TRUE` if `InitializeFromInitiateTaskId` was able to initialize the task properly using the submitted user name and task ID. `InitializeFromInitiateTaskId` returns `FALSE` if an incorrect user name and task ID are submitted or it was not able to execute.	

InitializeFromNextQueueTask

Method: InitializeFromNextQueueTask	

Method description: This method loads the next task from the task queue for the user in this task object.
 	
C# method call:
`public bool InitializeFromNextQueueTask(string strUserName, out string strError)`
 	
VB.NET method call:
`Public Function InitializeFromNextQueueTask(ByVal strUserName As String, ByRef strError As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` is the user's short name.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains an error message if the method does not successfully load the next task from the task queue.
Return Conditions	
`InitializeFromNextQueueTask` returns `TRUE` if `InitializeFromNextQueueTask` was able to execute. `InitializeFromNextQueueTask` returns `FALSE` if `InitializeFromNextQueueTask` could not execute or if there is no task in the queue.	

InitializeFromTaskId

Method: InitializeFromTaskID	

Method description: This method loads the task for the `strTaskID` passed to this object.
C# method call:
`public bool InitializeFromTaskId(string strTaskId)`
VB.NET method call:
`Public Function InitializeFromTaskId(ByVal strTaskId As String) As Boolean`	

Input	
`strTaskID`	**Input Type:** System.String **Description:** `strTaskID` is the unique task identifier to be loaded.
Return Conditions	
`InitializeFromTaskID` returns `TRUE` if `InitializeFromTaskID` was able to execute and was able to find the specified task. `InitializeFromTaskID` returns `FALSE` if `InitializeFromTaskID` was not able to execute or was not able to find the specified task. In the case that the past value for `strTaskID` is blank or empty, then `InitializeFromTaskID` returns an external exception which must be captured in the custom solution.	

InsertIntoRepeatedNode

Method: InsertIntoRepeatedNode	

Method description: This method inserts a node at a specified index of a repeated element.
C# method call:
`public bool InsertIntoRepeatedNode(string strRepeatedNodePath, long lIndex, out string strError)`
VB.NET method call:
`Public Function InsertIntoRepeatedNode(ByVal strRepeatedNodePath As String, ByVal lIndex As Long, ByRef strError As String) As Boolean`	

Input	
`strRepeatedNodePath`	**Input Type:** System.String **Description:** `strRepeatedNodePath` represents the complete path of the repeated node.
`lIndex`	**Input Type:** System.Int64 **Description:** `lIndex` represents where the node is to be added.

Method: InsertIntoRepeatedNode	
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to insert into a repeated node.
Return Conditions	
`InsertIntoRepeatedNode` returns `TRUE` if the node is added successfully. `InsertIntoRepeatedNode` returns `FALSE` if the node is not added successfully.	

PutTaskInQueue

Method: PutTaskInQueue	

Method description: This method puts the selected task back into the queue (if the task has been picked from the queue).
C# method call:
`public bool PutTaskInQueue(out string strError)`
VB.NET method call:
`Public Function PutTaskInQueue(ByRef strError As String) As Boolean`	

Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains an error message if the method does not place the selected task back into the queue.
Return Conditions	
`PutTaskInQueue` returns `TRUE` if `PutTaskInQueue` was able to execute. `PutTaskInQueue` returns `FALSE` if `PutTaskInQueue` could not execute or if the task is not a Queue task.	

Refresh

Method: Refresh	

Method description: This method saves the changes in memory made to the task data without the need for the EIK user to submit the task.
C# method call:
`public bool Refresh(out string strError)`
VB.NET method call:
`Public Function Refresh(ByRef strError As String) As Boolean`	

Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains an error message if the method does not refresh the task data at the Ultimus BPM Server.
Return Conditions	
`Refresh` returns `TRUE` if `Refresh` was able to refresh the task data. `Refresh` returns `FALSE` if `Refresh` could not refresh the task data.	

RemoveRepeatedNodeIndex

Method: RemoveRepeatedNodeIndex	

Method description: This method removes a specified node from a specified index of a repeated element.
C# method call:
`public bool RemoveRepeatedNodeIndex(string strRepeatedNodePath, long lIndex, out string strError)`
VB.NET method call:
`Public Function RemoveRepeatedNodeIndex(ByVal strRepeatedNodePath As String, ByVal lIndex As Long, ByRef strError As String) As Boolean`	

Input	
`strRepeatedNodePath`	**Input Type:** System.String **Description:** `strRepeatedNodePath` represents the complete path of the repeated node.
`lIndex`	**Input Type:** System.Int64 **Description:** `lIndex` represents from where the node is to be removed.

Method: RemoveRepeatedNodeIndex	
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to remove the node from the specified index.
Return Conditions	
`RemoveRepeatedNodeIndex` returns `TRUE` if the node at the specified index is removed successfully. `RemoveRepeatedNodeIndex` returns `FALSE` if the node at the specified index is not removed successfully.	

Return

Method: Return	

Method description: This method returns the task.
C# method call:
`public bool Return(string strMemo, string strSummary, bool bNoAbort, out string strError)`
VB.NET method call:
`Public Function [Return](ByVal strMemo As String, ByVal strSummary As String, ByVal bNoAbort As Boolean, ByRef strError As String) As Boolean`	

Input	
`strMemo`	**Input Type:** System.String **Description:** `strMemo` is the memo that goes with the task.
`strSummary`	**Input Type:** System.String **Description:** The incident summary for this incident can be set by specifying a value for `strSummary`.
`bNoAbort`	**Input Type:** System.Boolean **Description:** `bNoAbort` is a boolean value. If the value is `TRUE`, then the task cannot be aborted; if `FALSE`, then the task is abortable.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains an error message if the method does not return the task.

Method: Return
Return Conditions
`Return` returns `TRUE` if `Return` was able to execute. `Return` returns `FALSE` if `Return` could not execute or if it could not return the task successfully.
Example

```
'This code snippet is written in VB.NET
Dim oTask As New Ultimus.WFServer.Task
Dim strTaskId As String
Dim strError As String = ""

If oTask.InitializeFromTaskId(strTaskId) Then
   If oTask.Return("Memo details", "Summary", True, strError) Then
      MessageBox.Show("Task returned successfully")
   End If
End If
```

SaveTemplate

Method: SaveTemplate

Method description: This method saves the Begin step task as a template.
C# method call:
`public bool SaveTemplate(string strUser, string Summary, string strXML, out string strError)`
VB.NET method call:
`Public Function SaveTemplate(ByVal strUser As String, ByVal Summary As String, ByVal strXML As String, ByRef strError As String) As Boolean`

Input

`strUser`	**Input Type:** System.String **Description:** `strUser` represents the name of the user against which the template is to be saved.
`Summary`	**Input Type:** System.String **Description:** `Summary` represents the summary saved with the template.
`strXML`	**Input Type:** System.String **Description:** `strXML` represents the task data that is to be saved with the template.

Method: SaveTemplate	
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to save the task as a template.
Return Conditions	
`SaveTemplate` returns `TRUE` if the template is saved successfully. `SaveTemplate` returns `FALSE` if the template is not saved successfully.	

SaveXML

Method: SaveXML	

Method description: This method saves the updated Task XML data to Ultimus BPM Database.
C# method call:
`public bool SaveXML(out string strError)`
VB.NET method call:
`Public Function SaveXML(ByRef strError As String) As Boolean`	

Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to save the task XML data.
Return Conditions	
`SaveXML` returns `TRUE` if the XML data is saved successfully. `SaveXML` returns `FALSE` if the XML data is not saved successfully.	

Send

Method: Send	

Method description: This method submits the task.
 	
C# method call:
`public bool Send(string strMemo, string strSummary, bool bNoAbort, ref int nIncidentNo, out string strError)`
 	
VB.NET method call:
`Public Function Send(ByVal strMemo As String, ByVal strSummary As String, ByVal bNoAbort As Boolean, ByRef nIncidentNo As Integer, ByRef strError As String) As Boolean`	

Input	
`strMemo`	**Input Type:** System.String **Description:** `strMemo` is the memo that goes with the task.
`strSummary`	**Input Type:** System.String **Description:** The incident summary for this incident can be set by specifying a value for `strSummary`.
`bNoAbort`	**Input Type:** System.Boolean **Description:** `bNoAbort` is a boolean value. If the value is `TRUE`, then the task cannot be aborted; if `FALSE`, then the task is abortable.
Output	
`nIncidentNo`	**Output Type:** System.Int32 **Description:** `nIncidentNo` is the incident number assigned if this is a Begin step task.
`strError`	**Output Type:** System.String **Description:** `strError` contains an error message if the method does not submit the task and update the step's XML schema element values with new values.

Method: Send
Return Conditions
`Send` returns `TRUE` if `Send` was able to submit the task successfully. `Send` returns `FALSE` if `Send` could not execute or if the task could not be sent.
Comments
The parameter `nIncident` has different functionality, depending on particular conditions. These conditions are described below: • `0` input: If `0`or blank is used for `nIncident`, then the `Send` method is called; the task is sent as a transaction. In this scenario, the call does not return until the transaction completes. However, unlike the next condition, in this scenario the task's incident number is returned. • `-1` input: In case `-1` is input, the Ultimus EIK invokes the `Send` method to submit the task, and not have this call based on a transaction. This does not return the task's incident number.

SendFrom

Method: SendFrom	

Method description: In the event that the recipient of a step is a group, department, or JFG, this method can be used to explicitly send the task on behalf of the user contained in `strUserName`.
 	
C# method call:
`public bool SendFrom(string strUserName, string strMemo, string strSummary, bool bNoAbort, ref int nIncidentNo, out string strError)`
 	
VB.NET method call:
`Public Function SendFrom(ByVal strUserName As String, ByVal strMemo As String, ByVal strSummary As String, ByVal bNoAbort As Boolean, ByRef nIncidentNo As Integer, ByRef strError As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` is the user's short name.
`strMemo`	**Input Type:** System.String **Description:** `strMemo` is the memo that goes with the task.
`strSummary`	**Input Type:** System.String **Description:** The incident summary for this incident can be set by specifying a value for `strSummary`.

Method: SendFrom	
`bNoAbort`	**Input Type:** System.Boolean **Description:** `bNoAbort` is a boolean value. If the value is `TRUE`, then the task cannot be aborted; if `FALSE`, then the task is abortable.
Output	
`nIncidentNo`	**Output Type:** System.Int32 **Description:** `nIncidentNo` is the incident number assigned if this is a Begin step task.
`strError`	**Output Type:** System.String **Description:** `strError` contains an error message if the method does not submit the task and update the task XML schema element values with new values.
Return Conditions	
`SendFrom` returns `TRUE` if `SendFrom` was able to send the task successfully. `SendFrom` returns `FALSE` if `SendFrom` could not send the task.	
Comments	
The parameter `nIncident` has different functionality, depending on particular conditions. These conditions are described below: • `0` input: If `0`or blank is used for `nIncident`, then the `Send` method is called; the task is sent as a transaction. In this scenario, the call does not return until the transaction completes. However, unlike the next condition, in this scenario the task's incident number is returned. • `-1` input: In case `-1` is input, the Ultimus EIK invokes the `Send` method to submit the task, and not have this call based on a transaction. This does not return the task's incident number.	

SetNodeValue

Method: SetNodeValue	

Method description: This method sets the specified node value.
 	
C# method call:
`public bool SetNodeValue(string strNodeName, object objValue, out string strError)`
 	
VB.NET method call:
`Public Function SetNodeValue(ByVal strNodeName As String, ByVal objValue As Object, ByRef strError As String) As Boolean`	

Input	
`strNodeName`	**Input Type:** System.String **Description:** `strNodeName` represents the name of the node whose value is to be updated.
`objValue`	**Input Type:** System.Object **Description:** `objValue` represents the value with which a specified node's data is to be updated.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to set the specified node value.
Return Conditions	
`SetNodeValue` returns `TRUE` if the value is updated successfully. `SetNodeValue` returns `FALSE` if the value is not updated successfully.	

SetTaskXML

Method: SetTaskXML	

Method description: This method sets the specified task XML data.
 	
C# method call:
`public bool SetTaskXML(string strXML, bool bValidateXML, out string strError)`
 	
VB.NET method call:
`Public Function SetTaskXML(ByVal strXML As String, ByVal bValidateXML As Boolean, ByRef strError As String) As Boolean`	

Input	
`strXML`	**Input Type:** System.String **Description:** `strXML` represents the updated XML data.
`bValidateXML`	**Input Type:** System.Boolean **Description:** `bValidateXML` contains a Boolean value indicating whether to validate the specified XML. If set to `TRUE`, the XML is validated against the schema; if set to `FALSE`, the XML is not validated.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to set the task XML data.
Return Conditions	
`SetTaskXML` returns `TRUE` if the task XML data is updated successfully. `SetTaskXML` returns `FALSE` if the task XML data is not updated successfully.	

TakeBack

Method: TakeBack	

Method description: This method takes back a specified user's assigned task.
 	
C# method call:
`public bool TakeBack (string strUser)`
 	
VB.NET method call:
`Public Function TakeBack(ByVal strUser As String) As Boolean`	

Input	
`strUser`	**Input Type:** System.String **Description:** `strUser` is the user's short name for which a task is to be taken back.
Return Conditions	
`TakeBack` returns `TRUE` if `TakeBack` was able to take back a specified user's assigned task. `TakeBack` returns `FALSE` if there is not an assigned task to the specified user or if `TakeBack` could not execute.	

Methods in the `TaskList` class

The `TaskList` class represents a list of tasks. It can be initiated from a view, query to a user, any filter, or tasks that have been assigned by the user. Outlined below are the methods for the `TaskList` class (in alphabetical order):

- *GetAt*
- *GetFirstTask*
- *GetNextTask*
- *GetPrevTask*
- *GetTasksCount*
- *LoadAssignedTasks*
- *LoadFilteredTasks*
- *LoadQueueTasks*
- *LoadViewTasks*

GetAt

Method: GetAt	

Method description: This method retrieves the task at a specified position.
C# method call:
`public Ultimus.WFServer.Task GetAt(int position)`
VB.NET method call:
`Public Function GetAt(ByVal position As Integer) As Ultimus.WFServer.Task`	

Input	
`position`	**Input Type:** System.Int32 **Description:** `position` represents a numerical value of position/index where a task resides in a loaded `TaskList` instance.
Return Conditions	
`GetAt` returns an object of type `Task` if `GetAt` is able to get the task from the specified position. `GetAt` returns a null reference object if `GetAt` could not get the task at the specified position.	

GetFirstTask

Method: GetFirstTask

Method description: This method returns the first task in the loaded `TaskList` instance, and sets the internal counter to the first task in the list.
C# method call:
`public Ultimus.WFServer.Task GetFirstTask()`
VB.NET method call:
`Public Function GetFirstTask() As Ultimus.WFServer.Task`

Return Conditions
`GetFirstTask` returns an object of type `Task` if `GetFirstTask` is able to get the task from the first position. `GetFirstTask` returns a null reference object if `GetFirstTask` could not get the task at the first position.

GetNextTask

Method: GetNextTask

Method description: This method retrieves the next task from the task list.
C# method call:
`public Ultimus.WFServer.Task GetNextTask()`
VB.NET method call:
`Public Function GetNextTask() As Ultimus.WFServer.Task`

Return Conditions
`GetNextTask` returns an object of type `Task` if `GetNextTask` is able to get the task from the next position. `GetNextTask` returns a null reference object if `GetNextTask` is not able to get the next task.

GetPrevTask

Method: GetPrevTask

Method description: This method retrieves the previous task from the task list.
 	
C# method call:
`public Ultimus.WFServer.Task GetPrevTask()`
 	
VB.NET method call:
`Public Function GetPrevTask() As Ultimus.WFServer.Task`

Return Conditions
`GetNextTask` returns an object of type `Task` if `GetPrevTask` is able to get the task from the previous position. `GetPrevTask` returns a null reference object if `GetPrevTask` is not able to get the previous task.

GetTasksCount

Method: GetTasksCount

Method description: This method returns the number of tasks for the loaded `TaskList` instance.
C# method call:
`public int GetTasksCount()`
VB.NET method call:
`Public Function GetTasksCount() As Integer`

Return Conditions
`GetTasksCount` returns the number of tasks, as specified by the filter conditions.
Example

```
'This code snippet is written in VB.NET
Dim oFilter As New Ultimus.WFServer.TasklistFilter
Dim oTasklist As New Ultimus.WFServer.Tasklist
Dim EnumFilters As Ultimus.WFServer.Filters
Dim Icount As Integer
Dim ClientUser as String() = {"username"}

'return the number of current tasks for the specified user
oFilter.strArrUserName = ClientUser
oFilter.nFiltersMask = EnumFilters.nFilter_Current
oTasklist.LoadFilteredTasks(oFilter)
Icount = oTasklist.GetTasksCount()
MessageBox.Show(Icount)
```

LoadAssignedTasks

Method: LoadAssignedTasks	

Method description: This method loads the list of tasks assigned by the user.
 	
C# method call:
`public bool LoadAssignedTasks(string strUser, out string strErr)`
 	
VB.NET method call:
`Public Function LoadAssignedTasks(ByVal strUser As String, ByRef strErr As String) As Boolean`	

Input	
`strUser`	**Input Type:** System.String **Description:** `strUser` specifies the user's short name whose assigned tasks are to be loaded.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to load assigned tasks.
Return Conditions	
`LoadAssignedTasks` returns `TRUE` if `LoadAssignedTasks` was able to load the tasks assigned by the user. `LoadAssignedTasks` returns `FALSE` if there are no assigned tasks, or if `LoadAssignedTasks` could not load the tasks assigned by the user.	

LoadFilteredTasks

Method: LoadFilteredTasks	

Method description: This method loads the list of filtered tasks.
 	
C# method call:
`public bool LoadFilteredTasks(Ultimus.WFServer.TasklistFilter filter, out string strErr)`
 	
VB.NET method call:
`Public Function LoadFilteredTasks(ByVal filter As Ultimus.WFServer.TasklistFilter, ByRef strErr As String) As Boolean`	

Input	
`filter`	**Input Type:** instance of `TaskListFilter` class *Properties and methods in the `TaskListFilter` class* (outlined on page 104) **Description:** `filter` specifies which tasks are to be retrieved. A `TaskListFilter` object should be initialized as outlined in *Properties and methods in the `TaskListFilter` class* (page 104) and passed as a parameter for this call. The tasks are retrieved according to that filter.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to load filtered tasks.
Return Conditions	
`LoadFilteredTasks` returns `TRUE` if `LoadFilteredTasks` was able to load the task(s) based on the input filter. `LoadFilteredTasks` returns `FALSE` if there are no tasks based on the input filter, or if `LoadFilteredTasks` could not load the task(s).	
Example	

`'This code snippet is written in VB.NET`
`Dim m_TaskList As New Ultimus.WFServer.Tasklist`
`Dim m_Filter As New Ultimus.WFServer.TasklistFilter`
`m_Filter.strProcessNameFilter = "MyProcess"`
`m_Filter.nIncidentNo = "5"`
`m_TaskList.LoadFilteredTasks(m_Filter,strErr)`	

LoadQueueTasks

Method: LoadQueueTasks	

Method description: This method loads the list of queue tasks for a specified user.
 	
C# method call:
`public bool LoadQueueTasks(string strUser, out string strErr)`
 	
VB.NET method call:
`Public Function LoadQueueTasks(ByVal strUser As String, ByRef strErr As String) As Boolean`	

Input	
`strUser`	**Input Type:** System.String **Description:** `strUser` specifies the user's short name of the person whose queue tasks are to be loaded.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to load queued tasks.
Return Conditions	
`LoadQueueTasks` returns `TRUE` if `LoadQueueTasks` was able to execute and the specified user has queue tasks. `LoadQueueTasks` returns `FALSE` if `LoadQueueTasks` could not execute or the specified user does not have any queue tasks.	

LoadViewTasks

Method: LoadViewTasks	

Method description: This method loads the tasks for the specified user and the specified view.
 	
C# method call:
`public bool LoadViewTasks(string strUser, string strViewName, out string strErr)`
 	
VB.NET method call:
`Public Function LoadViewTasks(ByVal strUser As String, ByVal strViewName As String, ByRef strErr As String) As Boolean`	

Input	
`strUser`	**Input Type:** System.String **Description:** `strUser` specifies the user whose tasks are to be retrieved.
`strViewName`	**Input Type:** System.String **Description:** `strViewName` specifies the view name for tasks to be retrieved.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to load view tasks.
Return Conditions	
`LoadViewTasks` returns `TRUE` if `LoadViewTasks` was able to successfully load the specified task view. `LoadViewTasks` returns `FALSE` if `LoadViewTasks` could not load the specified task view.	

Properties and methods in the `TaskListFilter` class

The `TaskListListFilter` class describes a filter on a list of tasks. This class is used to initialize a task list and to define a user view. Outlined below are the properties and methods for the `TaskListFilter` class (in alphabetical order):

- *Properties for the `TaskListFilter` class*
- *AddProcess*
- *ClearDueDateFilters*
- *ClearStartDateFilters*
- *GetProcessAt*

- *GetProcessesCount*
- *RemoveAllProcesses*
- *RemoveProcessAt*
- *SetDateFilter*
- *SetDateFilterRelativeToToday*
- *SetDateFilterToSpecificDay*
- *SetDateFilterToToday*

Properties for the `TaskListFilter` class

Table 13. Properties for the `TaskListFilter` class

Property Name	Property Type and Description
`dEndDate`	**Type:** Double (date/time) **Description:** `dEndDate` represents tasks that end on a specific date.
`dStartDate`	**Type:** Double (date/time) **Description:** `dStartDate` filters on tasks that start on a specific date.
`dStartDate1`	**Type:** Double (date/time) **Description:** `dStartDate1` filters on tasks that start on a specific date when the `nStartDateType` is selected between specific dates.
`dStartDate2`	**Type:** Double (date/time) **Description:** `dStartDate2` filters on tasks that start on a specific date when the `nStartDateType` is selected between specific dates.
`nDateType`	**Type:** System.Int32 **Description:** `m_nDateType` acquires the date type for a specific client view. The following are the values returned by `nDateType`: **Return** — **Value** Any day — 0 Today — 1 On specific day — 2 In between specific day — 3 Relative to today — 4
`nEndIncidentNo`	**Type:** System.Int32 **Description:** `nEndIncidentNo` filters on the incident number of an Ultimus process. If called, all tasks until the last specified incident number are loaded.

Table 13. Properties for the `TaskListFilter` class *(Continued)*

Property Name	Property Type and Description
`nEndingDaysOfDueDate RelativeFromToday`	**Type:** System.Int32 **Description:** `nEndingDaysOfDueDateRelativeFromToday` returns the day ahead relative to today if the date type is set to "relative to today."
`nEndingDaysOfStartDate RelativeFromToday`	**Type:** System.Int32 **Description:** `nEndingDaysOfDueDateRelativeFromToday` returns the day ahead relative to today if the date type is set to "relative to today."

`nFiltersMask`	**Type:** System.Int32
	Description: `nFiltersMask` can be one of the values described in the *Enumerations for `WFServer.Filters` class* (outlined on page 27). When called, only the specified tasks are loaded.
	Example:
	`'This code snippet is written in VB.NET`
	`Dim m_TaskList As New Ultimus.WFServer.Tasklist`
	`Dim m_Filter As New Ultimus.WFServer.TasklistFilter`
	`Dim m_FilterEnum As Ultimus.WFServer.Filters`
	`m_Filter.strProcessNameFilter = "MyProcess"`
	`m_Filter.nFiltersMask = m_FilterEnum.nFilter_Urgent`
	`m_TaskList.LoadFilteredTasks(m_Filter)`

`nIncidentNo`	**Type:** System.Int32 **Description:** `nIncidentNo` filters on the incident number of an Ultimus process. If called, only tasks for the given incident number are loaded.
`nPriorityEnd`	**Type:** System.Int32 **Description:** `nPriorityEnd` filters on the priority of a task within an Ultimus process. Using both `nPriorityStart` and `nPriorityEnd` provide a filter on the priority of a task. When these are called, only tasks with priorities within the given range are loaded.
`nPriorityStart`	**Type:** System.Int32 **Description:** `nStartDateType` filters on the priority of a task within an Ultimus process. Using both `nPriorityStart` and `nPriorityEnd` provide a filter on the priority of a task. When these are called, only tasks with priorities within the given range are loaded.

Table 13. Properties for the `TaskListFilter` class *(Continued)*

Property Name	Property Type and Description
`nStartDateType`	**Type:** System.Int32 **Description:** `nStartDateType` acquires the date type based on when a task was activated for a specific client view. **Return** / **Value** Today / 1 Specific day / 2 In between specific dates / 3 (`StartDate1` and `StartDate2`) Start dates relative from today / 4 (`nEndingDaysOfStartDate RelativeFromToday` and `nStartingDaysOfStartDate RelativeFromToday`)
`nStartingDaysOfDueDate RelativeFromToday`	**Type:** System.Int32 **Description:** `nStartingDaysOfDueDateRelativeFromToday` specifies the tasks that were due a specified number of days prior to today if the date type is set to "relative to today."
`nStartingDaysOfStartDate RelativeFromToday`	**Type:** System.Int32 **Description:** `nStartingDaysOfStartDateRelativeFromToday` specifies the steps acativated a specified number of days prior to today if the date type is set to "relative to today."
`strArrUserName`	**Type:** array of System.String **Description:** `strArrUserName` filters on user name. The user can specify one or more user short names. When called, only tasks for the given user(s) are loaded.
`strProcessNameFilter`	**Type:** System.String **Description:** When `strProcessNameFilter` is called, only tasks for the given process name are loaded.
`strArrStepLabelFilter`	**Type:** array of System.String **Description:** `strArrStepLabelFilter` filters on a step label. When called, only tasks for the given step(s) are loaded.
`strSummaryFilter`	**Type:** System.String **Description:** `strSummaryFilter` filters on the incident summary summary of an incident. When called, only tasks whose current incident summary matches the given value are loaded.

AddProcess

Method: AddProcess	

Method description: This method adds a specified business process to the `TasklistFilter` class object.
 	
C# method call:
`public int AddProcess(string strProcessName, int lVersion)`
 	
VB.NET method call:
`Public Function AddProcess(ByVal strProcessName As String, ByVal lVersion As Integer) As Integer`
 	
Overload method calls
 	
C# method call:
`public int AddProcess(Ultimus.ClientServices.InstalledProcess process)`
 	
VB.NET method call:
`Public Function AddProcess(ByVal process As Ultimus.ClientServices.InstalledProcess) As Integer`	

Input	
`strProcessName`	**Input Type:** System.String **Description:** `strProcessName` represents the name of the business process which is to be added in the filter.
`lVersion`	**Input Type:** System.Int32 **Description:** `lVersion` represents the version of the specified business process.
`process`	**Input Type:** `Ultimus.ClientServices.InstalledProcess` **Description:** `process` represents the name of the business process which is to be added in the filter.
Return Conditions	
`AddProcess` returns `1` if the process is added successfully. `AddProcess` returns `0` if the process is not added successfully.	

ClearDueDateFilters

Method: ClearDueDateFilters

Method description: This method clears the date filters created with respect to due dates.
C# method call:
`public void ClearDueDateFilters()`
VB.NET method call:
`Public Sub ClearDueDateFilters()`

ClearStartDateFilters

Method: ClearStartDateFilters

Method description: This method clears the date filters created with respect to start dates.
C# method call:
`public void ClearStartDateFilters()`
VB.NET method call:
`Public Sub ClearStartDateFilters())`

GetProcessAt

Method: GetProcessAt	

Method description: This method retrieves the business process at the specified index.
C# method call:
`public Ultimus.ClientServices.InstalledProcess GetProcessAt(int index)`
VB.NET method call:
`Public Function GetProcessAt(ByVal index As Integer) As Ultimus.ClientServices.InstalledProcess`	

Input	
`Index`	**Input Type:** System.Int32 **Description:** `Index` represents the index number of the business process to be retrieved.
Return Conditions	
`GetProcessAt` returns the process in an object of `Ultimus.ClientServices.InstalledProcess` class.	

GetProcessesCount

Method: GetProcessesCount

Method description: This method retrieves the number of business processes added in the `TasklistFilter` class object.
C# method call:
`public int GetProcessesCount()`
VB.NET method call:
`Public Function GetProcessesCount() As Integer`

Return Conditions
`GetProcessesCount` returns the total number of business processes in a filter.

RemoveAllProcesses

Method: RemoveAllProcesses

Method description: This method removes all the business processes included in the filter.
C# method call:
`public void RemoveAllProcesses()`
VB.NET method call:
`Public Sub RemoveAllProcesses()`

RemoveProcessAt

Method: RemoveProcessAt	

Method description: This method removes a business process at a specified index.
C# method call:
`public void RemoveProcessAt(int index)`
VB.NET method call:
`Public Sub RemoveProcessAt(ByVal index As Integer)`	

Input	
`Index`	**Input Type:** System.Int32 **Description:** `Index` represents the index number of the business process that is to be removed.

SetDateFilter

Method: SetDateFilter	

Method description: This method sets the filter on the due date of the task. When called, only tasks having due dates between the `startDate` and `endDate` are loaded.
 	
C# method call:
`public void SetDateFilter(double startDate, double endDate)`
 	
VB.NET method call:
`Public Sub SetDateFilter(ByVal startDate As Double, ByVal endDate As Double)`	

Input	
`startDate`	**Input Type:** Double (date/time) **Description:** `startDate` represents the beginning due date of any task.
`endDate`	**Input Type:** Double (date/time) **Description:** `endDate` represents the ending due date of any task.

SetDateFilterRelativeToToday

Method: SetDateFilterRelativeToToday	

Method description: This method sets the date filter for tasks due in a time span relative to today. When called, tasks due on dates `nDaysAgo` before the current day and/or `nDaysAfter` (after the current day) are loaded.
 	
C# method call:
`public void SetDateFilterRelativeToToday(int nDaysAgo, int nDaysAfter)`
 	
VB.NET method call:
`Public Sub SetDateFilterRelativeToToday(ByVal nDaysAgo As Integer, ByVal nDaysAfter As Integer)`	

Input	
`nDaysAgo`	**Input Type:** System.Int32 **Description:** `nDaysAgo` represents the number of days prior to the current day tasks are to be loaded.
`nDaysAfter`	**Input Type:** System.Int32 **Description:** `nDaysAfter` represents the number of days after the current day tasks are to be loaded.

SetDateFilterToSpecificDay

Method: SetDateFilterToSpecificDay	

Method description: This method sets the date filter for tasks due on a specified due date. When called, only tasks having a due date specified are loaded.
 	
C# method call:
`public void SetDateFilterToSpecificDay(double dtSpecificDay)`
 	
VB.NET method call:
`Public Sub SetDateFilterToSpecificDay(ByVal dtSpecificDay As Double)`	

Input	
`dtSpecificDay`	**Input Type:** Double (date/time) **Description:** `dtSpecificDay` represents a specific date that tasks are to be loaded.

SetDateFilterToToday

Method: SetDateFilterToToday

Method description: This method sets the filter for tasks due on the current day only. When called, only tasks due on the current date are loaded.
 	
C# method call:
`public void SetDateFilterToToday()`
 	
VB.NET method call:
`Public Sub SetDateFilterToToday()`

The Ultimus EIK interface: classes within the OC namespace

OC namespace contains five classes (shown in Figure 4).

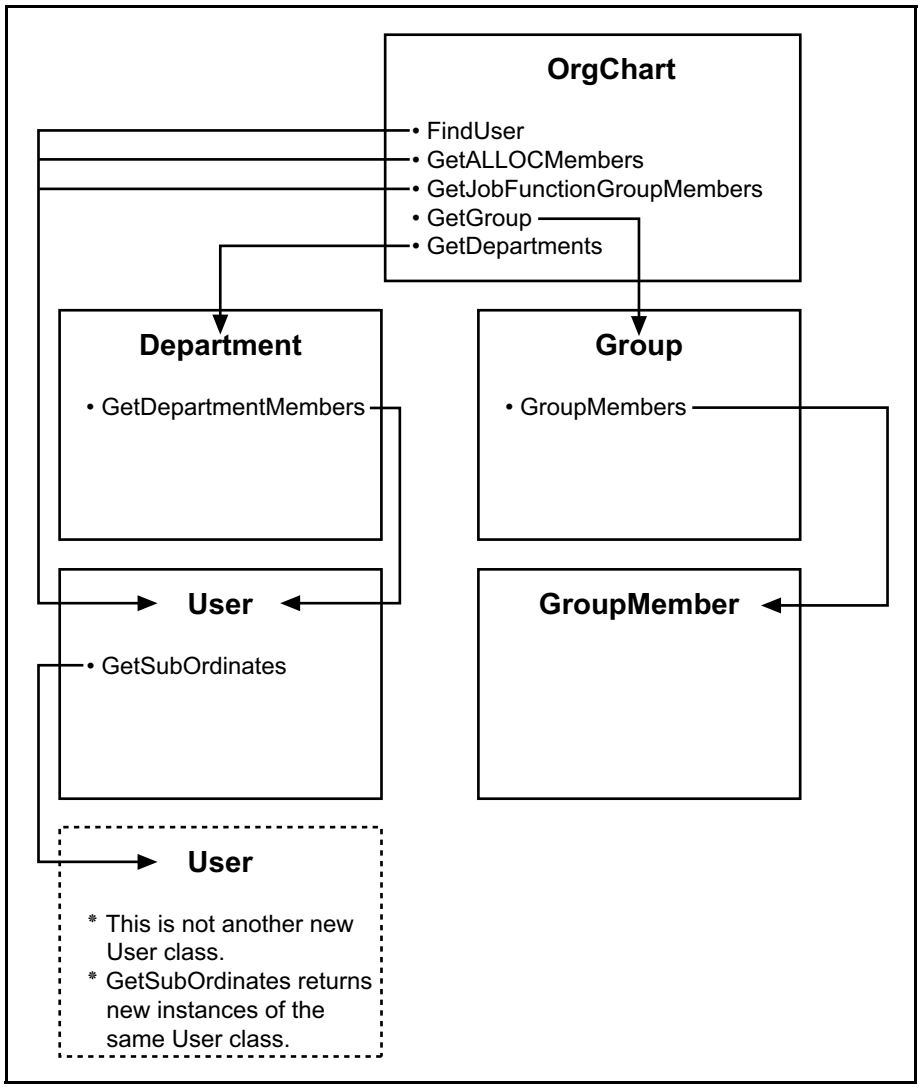

Figure 4. An architectural schematic for the OC namespace

Each class contains methods and properties (shown in Figure 5).

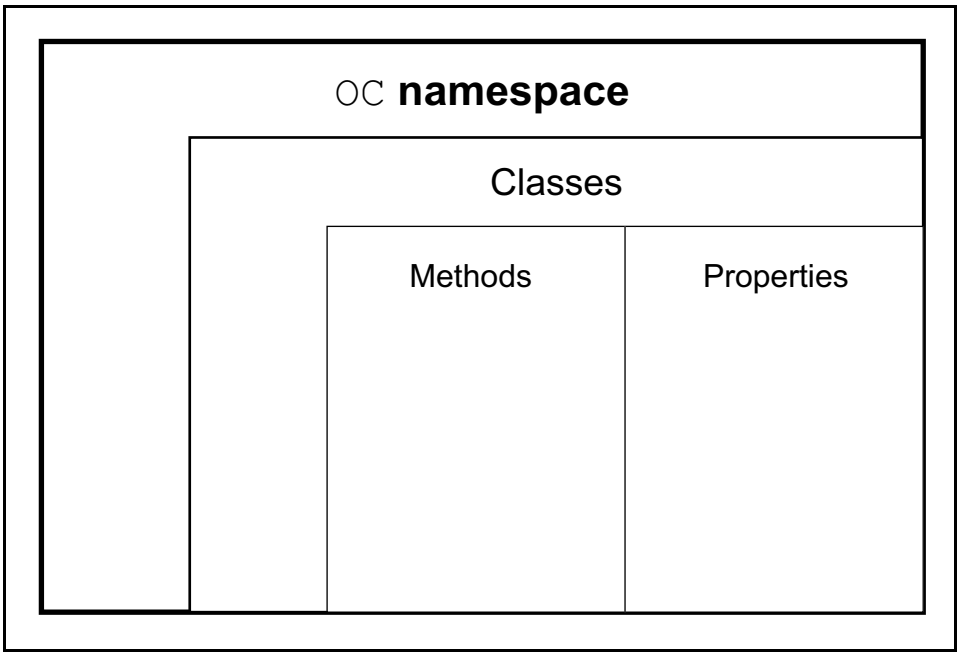

Figure 5. The Ultimus EIK architectural relationship

Outlined in the tables below are descriptions of all status codes (enumerations), properties and methods within each class of the `OC` namespace. This section contains:

- *Enumerations (ENUMs)*
- *Properties and methods in the `Department` class*
- *Properties and method in the `Group` class*
- *Properties in the `GroupMember` class*
- *Methods in the `OrgChart` class*
- *Properties and methods in the `User` class*

Enumerations (ENUMs)

The following table outlines status codes pertaining to user rights. These are defined by the enumerations (ENUMs) in `OC.UserRights` class of the `ultikstructures.dll` file.

Table 14. ENUMs for `OC.UserRights` class

Enumeration	Description
`CanConfigureViews`	User rights to configure views
`Director`	User rights to run Ultimus Director
`FlostationConfiguration`	User rights to run Ultimus Flostation Configuration
`FullAccess`	Full access to run all modules
`None`	No user rights
`OrganizationChart`	User rights to run Ultimus Organization Chart
`ProcessAdministration`	User rights to run Ultimus Process Administrator
`Reports`	User rights to run Ultimus Reports
`SystemAdministration`	User rights to run Ultimus System Administrator

Properties and methods in the `Department` class

The `Department` class describes a department within the Ultimus Organization Charts module.

`Ultimus.OC.OrgChart` class has two constructors used in the following ways:

- Use this constructor without any parameter. Ultimus default Business Organization is used. Below is an example:

  ```
  Ultimus.OC.OrgChart oc = new OrgChart()
  ```

- Use this constructor with a parameter which specifies an Organization name. Below is an example:

  ```
  Ultimus.OC.OrgChart oc = new OrgChart(<Organization name>)
  ```

  where *`<Organization name>`* represents the name of the queried organization

Outlined below are the properties and methods for the `Department` class (in alphabetical order):

- *Properties for the `Department` class*
- *GetDepartmentJobFunctionGroups*
- *GetDepartmentManager*
- *GetSubDepartments*

Properties for the Department class

Table 15. Properties for the Department class

Property Name	Property Type and Description
strDepartmentID	**Type:** System.String **Description:** strDepartmentID provides a unique identifier for this department.
strDepartmentName	**Type:** System.String **Description:** strDepartmentName represents the name of the department.
strOrganizationName	**Type:** System.String **Description:** strOrganizationName represents the name of the organization.

GetDepartmentJobFunctionGroups

Method: GetDepartmentJobFunctionGroups	

Method description: This method acquires all of the job function groups in the department.
C# method call:
`public bool GetDepartmentJobFunctionGroups(out string[] strDeptJobFnGpsArr)`
VB.NET method call:
`Public Function GetDepartmentJobFunctionGroups(ByRef strDeptJobFnGpsArr() As String) As Boolean`	

Output	
strDeptJobFnGpsArr	**Output Type:** System.String **Description:** strDeptJobFnGpsArr returns all the job function groups in the department.
Return Conditions	
GetDepartmentJobFunctionGroups returns TRUE if GetDepartmentJobFunctionGroups was able to execute. GetDepartmentJobFunctionGroups returns FALSE if GetDepartmentJobFunctionGroups could not execute.	

GetDepartmentManager

Method: GetDepartmentManager	

Method description: This method returns the department manager as a single user.
C# method call:
`public bool GetDepartmentManager(out User user)`
VB.NET method call:
`Public Function GetDepartmentManager(ByRef user As Ultimus.OC.User) As Boolean`	

Output	
`user`	**Output Type:** Object of `Ultimus.OC.User` class **Description:** `user` represents the manager of the specified department.
Return Conditions	
`GetDepartmentManager` returns `TRUE` if the manager is returned successfully. `GetDepartmentManager` returns `FALSE` if `GetDepartmentManager` could not execute.	

GetDepartmentMembers

Method: GetDepartmentMembers	

Method description: This method returns all the members of the department.
C# method call:
`public bool GetDepartmentMembers(out Ultimus.OC.User[] UserList)`
VB.NET method call:
`Public Function GetDepartmentMembers(ByRef UserList() As Ultimus.OC.User) As Boolean`	

Output	
`UserList`	**Output Type:** instance of *Properties and methods in the `User` class* (outlined on page 131) **Description:** `UserList` returns all the members of the department.
Return Conditions	
`GetDepartmentMembers` returns `TRUE` if `GetDepartmentMembers` was able to execute. `GetDepartmentMembers` returns `FALSE` if `GetDepartmentMembers` could not execute.	

GetSubDepartments

Method: GetSubDepartments	

Method description: This method return all the sub departments present in a specified department.
 	
C# method call:
`public bool GetSubDepartments(out Ultimus.OC.Department[] subDepts)`
 	
VB.NET method call:
`Public Function GetSubDepartments(ByRef subDepts() As Ultimus.OC.Department) As Boolean`	

Output	
`subDepts`	**Output Type:** Ultimus.OC.Department **Description:** `subDepts` contains all the sub departments present in a specified department.
Return Conditions	
`GetSubDepartments` returns `TRUE` if `GetSubDepartments` is able to execute successfully. `GetSubDepartments` returns `FALSE` if `GetSubDepartments` is unable to execute successfully.	

Properties and method in the `Group` class

The `Group` class offers a class description.

`Ultimus.OC.Group` class has two constructors used in the following ways:

- Use this constructor without any parameter. Ultimus default Business Organization is used. Below is an example:

  ```
  Ultimus.OC.Group grp = new Group()
  ```

- Use this constructor with a parameter which specifies a organization name. Below is an example:

  ```
  Ultimus.OC.Group grp = new Group(<Organization name>)
  ```

  where *`<Organization name>`* represents the name of the queried organization

Outlined below are the properties and method for the `Group` class (in alphabetical order):

- *Properties for the `Group` class*
- *IsMemberOfGroup*

Properties for the Group class

Table 16. Properties for the Group class

Property Name	Property Type and Description

GroupMembers	**Type:** GroupMember
	Description: GroupMembers lists the users included in the group.
	Example:
	`'This code snippet is written in VB.NET`
	`Dim UltOrgChart As New Ultimus.OC.OrgChart`
	`Dim ReturnedGroup as Ultimus.OC.Group`
	`Dim NumGroupMembers as Integer`
	`UltOrgChart.GetGroup(TextBox3.Text, ReturnedGroup)`
	`NumGroupMembers = ReturnedGroup.GroupMembers.Length`
	`For xx = 0 To NumGroupMembers - 1`
	`ListBox2.Items.Add(ReturnedGroup.GroupMembers(xx).strFullName)`
	`ListBox3.Items.Add(ReturnedGroup.GroupMembers(xx).strMemberName)`
	`Next xx`

strJobId	**Type:** System.String **Description:** strJobID represents the ID of the group.
strGroupName	**Type:** System.String **Description:** strGroupName represents the name of the group.
strGroupProfile	**Type:** System.String **Description:** strGroupProfile pertains to the group which has rights to modify this group.

IsMemberOfGroup

Method: IsMemberOfGroup	
Method description: This method returns TRUE if strUserName is a member of the group; otherwise, it returns FALSE. **C# method call:** public bool IsMemberOfGroup(string strUserName) **VB.NET method call:** Public Function IsMemberOfGroup(ByVal strUserName As String) As Boolean	
Input	
strUserName	**Input Type:** System.String **Description:** strUserName is the user short name (or log on ID) of a user for which group membership is being queried. The format for strUserName should be domain_name/user_short_name.
Return Conditions	
IsMemberOfGroup returns TRUE if the specified user is a member of the specified group. IsMemberOfGroup returns FALSE if the specified user is not a member of the specified group.	

Properties in the GroupMember class

The GroupMember class defines a member of a group. Outlined below are the properties for the GroupMember class (in alphabetical order).

Table 17. Properties for the GroupMember class

Property Name	Property Type and Description
NSequence	**Type:** System.Int32 **Description:** Used for sequential groups, NSequence represents the order number of the user.
NWeight	**Type:** System.Int32 **Description:** If this is a weighted group, NWeight represents the weight value.
strFullName	**Type:** System.String **Description:** strFullName represents the full name of the user.

Table 17. Properties for the GroupMember class *(Continued)*

Property Name	Property Type and Description
strGroupName	**Type:** System.String **Description:** strGroupName represents the group name with which the member belongs.
strMemberName	**Type:** System.String **Description:** strMemberName represents the name of the group member.

Methods in the OrgChart class

The OrgChart class provides the basic organizational chart functions for the Ultimus Organization Charts module.

Ultimus.OC.OrgChart class has two constructors used in the following ways:

- Use this constructor without any parameter. Ultimus default Business Organization is used. Below is an example:

  ```
  Ultimus.OC.OrgChart oc = new OrgChart()
  ```

- Use this constructor with a parameter which specifies an Organization name. Below is an example:

  ```
  Ultimus.OC.OrgChart oc = new OrgChart(<Organization name>)
  ```

  where *<Organization name>* represents the name of the queried organization

Outlined below are the methods for the OrgChart class (in alphabetical order):

- *FindDepartment*
- *FindUser*
- *GetAllOCMembers*
- *GetDepartments*
- *GetGroup*
- *GetGroupNames*
- *GetJobFunctionGroupMembers*
- *GetTopLevelDepartments*
- *VerifyUser*

FindDepartment

Method: FindDepartment	

Method description: This method loads a particular department by name or ID.
 	
C# method call:
`public bool FindDepartment (string strDeptName, string strDeptId, out Ultimus.OC.Department Dept)`
 	
VB.NET method call:
`Public Function FindDepartment(ByVal strDeptName As String, ByVal strDeptId As String, ByRef Dept As Ultimus.OC.Department) As Boolean`	

Input	
`strDeptName`	**Input Type:** System.String **Description:** `strDeptName` is the department name that is to be loaded.
`strDeptId`	**Input Type:** System.String **Description:** `strDeptID` is the department ID that is to be loaded.
Output	
`Dept`	**Output Type:** instance of the *Properties and methods in the `Department` class* (outlined on page 115) **Description:** `Dept` returns the department queried.
Return Conditions	
`FindDepartment` returns `TRUE` if `FindDepartment` was able to execute. `FindDepartment` returns `FALSE` if `FindDepartment` could not execute.	
Example	

```
'This code snippet is written in VB.NET
Dim ultOrgChart As New Ultimus.OC.OrgChart
Dim dept As Ultimus.OC.Department
If (ultOrgChart.FindDepartment("Dept Name", "Dept ID", dept)) Then
   MessageBox.Show("Method executed successfully.")
End If
```

FindUser

Method: FindUser	

Method description: This method finds a user in Ultimus Organization Charts and returns their information. This method searches according to the parameters provided. (All input parameters are optional, but at least one must be provided for a successful search.)
 	
C# method call:
`public bool OrgChart.FindUser(string strShortName, string strFullName, string strUserId, out Ultimus.OC.User user)`
 	
VB.NET method call:
`Public Function FindUser(ByVal strShortName As String, ByVal strFullName As String, ByVal strUserId As String, ByRef user As Ultimus.OC.User) As Boolean`	

Input	
`strShortName`	**Input Type:** System.String **Description:** `strShortName` is the user short name (or log on ID) of a user that is being queried. The format for `strShortName` should be `domain_name/user_short_name`.
`strFullName`	**Input Type:** System.String **Description:** `strFullName` is the user's full name that is being queried.
`strUserId`	**Input Type:** System.String **Description:** `strUserId` represents the unique identifier of the user.
Output	
`user`	**Output Type:** instance of the *Properties and methods in the `User` class* (outlined on page 131) **Description:** `user` returns the user queried.
Return Conditions	
`FindUser` returns `TRUE` if `FindUser` was able to execute. `FindUser` returns `FALSE` if `FindUser` could not execute.	

Method: FindUser
Example

```
'This code snippet is written in VB.NET
Dim UltOrgChart As New Ultimus.OC.OrgChart
Dim InputShortName as string
Dim InputFullName as string
Dim InputUserId as string
Dim FoundUser as Ultimus.OC.User
InputShortName = "myshortname"
InputFullName = ""
InputUserId = ""
UltOrgChart.FindUser(InputShortName,InputFullName,InputUserId,FoundUser)
```

GetAllOCMembers

Method: GetAllOCMembers	

Method description: This method acquires a list of all the members in the Ultimus Organization Charts module.
C# method call:
`public bool GetAllOCMembers(out Ultimus.OC.User[] UserList)`
VB.NET method call:
`Public Function GetAllOCMembers(ByRef UserList() As Ultimus.OC.User) As Boolean`	

Output	
`UserList`	**Output Type:** instance of the *Properties and methods in the `User` class* (outlined on page 131) **Description:** `UserList` returns a list of all members in the Ultimus Organization Charts module.
Return Conditions	
`GetAllOCMembers` returns `TRUE` if `GetAllOCMembers` was able to execute. `GetAllOCMembers` returns `FALSE` if `GetAllOCMembers` could not execute.	
Example	

```
'This code snippet is written in VB.NET
'AllUsers is an array of OC.User classes
Dim UltOrgChart As New Ultimus.OC.OrgChart
Dim AllUsers() as Ultimus.OC.User
UltOrgChart.GetAllOCMembers(AllUsers)
```

GetDepartments

Method: GetDepartments	

Method description: This method acquires a list of all the departments within an Ultimus Organization Charts chart.
 	
C# method call:
`public bool GetDepartments(out Ultimus.OC.Department[] DeptList)`
 	
VB.NET method call:
`Public Function GetDepartments(ByRef DeptList() As Ultimus.OC.Department) As Boolean`	

Output	
`DeptList`	**Output Type:** instance of the *Properties and methods in the `Department` class* (outlined on page 115) **Description:** `DeptList` returns a list of all departments within an Ultimus Organization Charts organization.
Return Conditions	
`GetDepartments` returns `TRUE` if `GetDepartments` was able to execute. `GetDepartments` returns `FALSE` if `GetDepartments` could not execute.	
Example	

```
'This code snippet is written in VB.NET
'Depts is an array of OC.Department classes
Dim UltOrgChart As New Ultimus.OC.OrgChart
Dim Depts() As Ultimus.OC.Department
UltOrgChart.GetDepartments(Depts)
```

GetGroup

Method: GetGroup	

Method description: This method returns the group specified by group name.
 	
C# method call:
`public bool GetGroup(string strGroupName, out Ultimus.OC.Group group)`
 	
VB.NET method call:
`Public Function GetGroup(ByVal strGroupName As String, ByRef group As Ultimus.OC.Group) As Boolean`	

Input	
`strGroupName`	**Input Type:** System.String **Description:** `strGroupName` is the group name (or log on ID) that is being queried.
Output	
`group`	**Output Type:** instance of the *Properties and method in the `Group` class* (outlined on page 118) **Description:** `group` returns the queried group by group name.
Return Conditions	
`GetGroup` returns `TRUE` if `GetGroup` was able to execute and an Ultimus Group exists for the specified group name. `GetGroup` returns `FALSE` if `GetGroup` was not able to execute or no Ultimus Group exists for the specified group name.	
Example	

```
'This code snippet is written in VB.NET
Dim UltOrgChart As New Ultimus.OC.OrgChart
Dim ReturnedGroup as New Ultimus.OC.Group
Dim SuppliedGroupName as String
SuppliedGroupName = "AnyGroup"
ReturnedGroup = UltOrgChart.GetGroup(SuppliedGroupName,ReturnedGroup)
```

GetGroupNames

Method: GetGroupNames	

Method description: This method acquires a list of groups in the Ultimus Organization Charts module.
 	
C# method call:
`public bool GetGroupNames(out string[] strGroupArr)`
 	
VB.NET method call:
`Public Function GetGroupNames(ByRef strGroupArr() As String) As Boolean`	

Output	
`strGroupArr`	**Output Type:** System.String **Description:** `strGroupsArr` returns a list of all groups within an Ultimus Organization Charts organization.
Return Conditions	
`GetGroupNames` returns `TRUE` if `GetGroupNames` was able to execute. `GetGroupNames` returns `FALSE` if `GetGroupNames` could not execute.	
Example	

```
'This code snippet is written in VB.NET
Dim UltOrgChart As New Ultimus.OC.OrgChart
Dim RetGroupList() As String
UltOrgChart.GetGroupNames(RetGroupList)
```

GetJobFunctionGroupMembers

Method: GetJobFunctionGroupMembers	

Method description: This method acquires a list of all members within a job function group.
 	
C# method call:
`public bool GetJobFunctionGroupMembers(string strJobFunctionGroup, out Ultimus.OC.User[] UserList)`
 	
VB.NET method call:
`Public Function GetJobFunctionGroupMembers(ByVal strJobFunctionGroup As String, ByRef UserList() As Ultimus.OC.User) As Boolean`	

Input	
`strJobFunctionGroup`	**Input Type:** System.String **Description:** `strJobFunctionGroup` is the job function name that is being queried.
Output	
`UserList`	**Output Type:** instance of the *Properties and methods in the `User` class* (outlined on page 131) **Description:** `UserList` returns a list of all members within a job function inside an Ultimus Organization Charts organization.
Return Conditions	
`GetJobFunctionGroupMembers` returns `TRUE` if `GetJobFunctionGroupMembers` was able to execute. `GetJobFunctionGroupMembers` returns `FALSE` if `GetJobFunctionGroupMembers` could not execute.	
Example	

```
'This code snippet is written in VB.NET
Dim UltOrgChart As New Ultimus.OC.OrgChart
Dim ReturnedUsers() As Ultimus.OC.User
Dim SuppliedJFGName as String
'JFGs are stored internally as Chart*JFGName
'But for this method call, only pass the JFGName, do not pass chart*
SuppliedJFGName = "AnyJFGName"
UltOrgChart.GetJobFunctionGroupMembers(SuppliedJFGName, ReturnedUsers)
```

GetTopLevelDepartments

Method: GetTopLevelDepartments	

Method description: This method loads the list of top-level departments (excluding sub-charts).
 	
C# method call:
`public bool GetTopLevelDepartments(out Ultimus.OC.Department[] DeptList)`
 	
VB.NET method call:
`Public Function GetTopLevelDepartments(ByRef DeptList() As Ultimus.OC.Department) As Boolean`	

Output	
`DeptList`	**Output Type:** array of instance of the *Properties and methods in the `Department` class* (outlined on page 115) **Description:** `DeptList` returns a list of all top-level departments; this list does not include any sub-charts.
Return Conditions	
`GetTopLevelDepartments` returns `TRUE` if `GetTopLevelDepartments` was able to execute. `GetTopLevelDepartments` returns `FALSE` if `GetTopLevelDepartments` could not execute.	
Example	

```
'This code snippet is written in VB.NET
Dim ultOrgChart As New Ultimus.OC.OrgChart
    Dim dept As Ultimus.OC.Department()
    If (ultOrgChart.GetTopLevelDepartments(dept)) Then
        If (dept.Length > 0) Then
            MessageBox.Show("Dept Exist")
        End If
    End If
```

VerifyUser

Method: VerifyUser	

Method description: This method verifies a user's log on user name and password to be valid. (This method returns TRUE if the user name/password combination is valid.)
 	
C# method call:
`public bool VerifyUser(string strUser, string strPassword)`
 	
VB.NET method call:
`Public Function VerifyUser(ByVal strUser As String, ByVal strPassword As String) As Boolean`	

Input	
`strUser`	**Input Type:** System.String **Description:** `strUser` is the log on user name of the user to be verified. The format for `strUser` should be `domain_name/user_short_name`.
`strPassword`	**Input Type:** System.String **Description:** `strPassword` is the password of the user to be verified.
Return Conditions	
`VerifyUser` returns `TRUE` if the input user name and password is correct for the specified user. `VerifyUser` returns `FALSE` if the input user name and password is not correct for the specified user.	

Method: VerifyUser
Example

```
'This code snippet is written in VB.NET
Dim UltOrgChart As New Ultimus.OC.OrgChart
Dim Uid As String
Dim Pswd As String
Dim ReturnFlag As Boolean
Uid = "anyname"
Pswd = "anypassword"
ReturnFlag = UltOrgChart.VerifyUser(Uid, Pswd)
If ReturnFlag = True Then
'entered password is correct
End If
```

Properties and methods in the `User` class

The `User` class describes a user within the Ultimus Organization Charts module.

`Ultimus.OC.User` class has two constructors used in the following ways:

- Use this constructor without any parameter. Ultimus default Business Organization is used. Below is an example:

  ```
  Ultimus.OC.User usr = new User()
  ```

- Use this constructor with a parameter which specifies an Organization name. Below is an example:

  ```
  Ultimus.OC.User usr = new User(<Organization name>)
  ```

  where *<Organization name>* represents the name of the queried organization

Outlined below are the properties and methods for the `User` class (in alphabetical order):

- *Properties for the `User` class*
- *AssignAllCurrentTasks*
- *AssignAllFutureTasks*
- *GetAssociates*
- *GetDepartment*
- *GetExactCaseUser*
- *GetManager*
- *GetManagerInDepartment*
- *GetOCSubOrdinates*

- *GetSubordinates*
- *GetSupervisor*
- *GetSupervisorInDepartment*
- *GetSupervisorOfJobFunction*
- *GetUserDepartments*
- *GetUserGroups*
- *GetUserGroups*
- *GetUserJFGs*
- *GetUserPrefs*
- *GetUserRights*
- *GetViewList*
- *IsJFG*
- *IsMemberOfGroup*
- *SetAssociates*
- *SetUserPrefs*

Properties for the `User` class

Table 18. Properties for the `User` class

Property Name	Property Type and Description
`strDepartmentName`	**Type:** System.String **Description:** `strDepartmentName` represents the department to which the user belongs.
`strJobFunction`	**Type:** System.String **Description:** `strJobFunction` represents the job function of the user.
`strUserFullName`	**Type:** System.String **Description:** `strUserFullName` represents the user's full name.
`strUserID`	**Type:** System.String **Description:** `strUserId` represents a unique identifier for the user.
`strUserName`	**Type:** System.String **Description:** `strUserName` represents the user's short name (or log on ID).

AssignAllCurrentTasks

Method: AssignAllCurrentTasks	

Method description: This method assigns all current tasks for this user to a specified user.
C# method call:
`public bool AssignAllCurrentTasks(string strAssignedTo)`
VB.NET method call:
`Public Function AssignAllCurrentTasks(ByVal strAssignedTo As String) As Boolean`	

Input	
`strAssignedTo`	**Input Type:** System.String **Description:** `strAssignedTo` specifies to which user all current tasks are to be assigned.
Return Conditions	
`AssignAllCurrentTasks` returns `TRUE` if `AssignAllCurrentTasks` was able to execute. `AssignAllCurrentTasks` returns `FALSE` if `AssignAllCurrentTasks` could not execute.	

AssignAllFutureTasks

Method: AssignAllFutureTasks	

Method description: This method assigns all future tasks for this user to a specified user until a specific date.
 	
C# method call:
`public bool AssignAllFutureTasks(string strAssignedTo, double DateAssignedUntil)`
 	
VB.NET method call:
`Public Function AssignAllFutureTasks(ByVal strAssignedTo As String, ByVal DateAssignedUntil As double) As Boolean`	

Input	
`strAssignedTo`	**Input Type:** System.String **Description:** `strAssignedTo` specifies to which user all future tasks are to be assigned.
`DateAssignedUntil`	**Input Type:** Double **Description:** `DateAssignedUntil` specifies at which date all future tasks are to be assigned.
Return Conditions	
`AssignAllFutureTasks` returns `TRUE` if `AssignAllFutureTasks` was able to execute. `AssignAllFutureTasks` returns `FALSE` if `AssignAllFutureTasks` could not execute.	

GetAssociates

Method: GetAssociates	

Method description: This method retrieves user associates.
 	
C# method call:
`public bool GetAssociates(out Ultimus.ClientServices.UserAssociate[] listUserAssociates)`
 	
VB.NET method call:
`Public Function GetAssociates(ByRef listUserAssociates() As Ultimus.ClientServices.UserAssociate) As Boolean`	

Output	
`listUserAssociates`	**Output Type:** An object array of `Ultimus.ClientServices.UserAssociate` class **Description:** `listUserAssociates` represents the specified user's associates.
Return Conditions	
`GetAssociates` returns `TRUE` if the associates are returned successfully. `GetAssociates` returns `FALSE` if the associates are not returned successfully.	

GetDepartment

Method: GetDepartment	

Method description: This method acquires the department to which the user belongs.
 	
C# method call:
`public bool GetDepartment(out string strDepartment)`
 	
VB.NET method call:
`Public Function GetDepartment(ByRef strDepartment As String) As Boolean`	

Output	
`strDepartment`	**Output Type:** System.String **Description:** `strDepartment` returns the department to which the user belongs.
Return Conditions	
`GetDepartment` returns `TRUE` if `GetDepartment` was able to execute. `GetDepartment` returns `FALSE` if `GetDepartment` could not execute.	

GetExactCaseUser

Method: GetExactCaseUser	

Method description: `strExactUser` returns the exact case of the user's short name, as specified in Ultimus Organization Charts, if the value of `strUserName` is a valid Ultimus Organization Charts user. In the case that `strUserName` is a value not in Ultimus Organization Charts, then `strExactUser` simply returns the same value as the supplied `strUserName` value.
 	
C# method call:
`public bool GetExactCaseUser(string strUserName, out string strExactUser)`
 	
VB.NET method call:
`Public Function GetExactCaseUser(ByVal strUserName As String, ByRef strExactUser As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the user's short name (or log on ID). The format for `strUserName` should be `domain_name/user_short_name`.

Method: GetExactCaseUser	
Output	
strExactUser	**Output Type:** System.String **Description:** strExactUser represents the user's short name value, as stored in Ultimus Organization Charts.
Return Conditions	
GetExactCaseUser returns TRUE if GetExactCaseUser was able to execute. GetExactCaseUser returns FALSE if GetExactCaseUser could not execute.	

GetManager

Method: GetManager	

Method description: This method acquires the manager of the user.
C# method call:
`public bool GetManager(out string strManager)`
VB.NET method call:
`Public Function GetManager(ByRef strManager As String) As Boolean`	

Output	
strManager	**Output Type:** System.String **Description:** strManager returns the manager of the user.
Return Conditions	
GetManager returns TRUE if GetManager was able to execute. GetManager returns FALSE if GetManager could not execute.	

GetManagerInDepartment

Method: GetManagerInDepartment	

Method description: This method acquires the manager of a specified department.
 	
C# method call:
`public bool GetManagerInDepartment(string strDepartment, out string strManager)`
 	
VB.NET method call:
`Public Function GetManagerInDepartment(ByVal strDepartment As String, ByRef strManager As String) As Boolean`	

Input	
`strDepartment`	**Input Type:** System.String **Description:** `strDepartment` represents the name of the department whose manager is to be retrieved.
Output	
`strManager`	**Output Type:** System.String **Description:** `strManager` represents the name of the manager of the specified department.
Return Conditions	
`GetManagerInDepartment` returns `TRUE` if `GetManagerInDepartment` was able to execute successfully. `GetManagerInDepartment` returns `FALSE` if `GetManagerInDepartment` could not execute.	

GetOCSubOrdinates

Method: GetOCSubOrdinates	

Method description: This method acquires the subordinates of the user, including users, job functions, and departments, and other attributes.
 	
C# method call:
`public bool GetOCSubOrdinates(out System.Collections.ArrayList lstSubOrdinates)`
 	
VB.NET method call:
`Public Function GetOCSubOrdinates(ByRef lstSubOrdinates As System.Collections.ArrayList) As Boolean`	

Output	
`lstSubOrdinates`	**Output Type:** System.Collections.ArrayList **Description:** `lstSubOrdinates`, an array list, contains all the subordinates of the user.
Return Conditions	
`GetOCSubOrdinates` returns `TRUE` if `GetOCSubOrdinates` is able to execute successfully. `GetOCSubOrdinates` returns `FALSE` if `GetOCSubOrdinates` is unable to execute successfully.	

Method: GetOCSubOrdinates

Example

```
'This code snippet is written in VB.NET
   Dim UltOrgChart As New Ultimus.OC.OrgChart
   Dim InputShortName As String
   Dim InputFullName As String
   Dim InputUserId As Integer
   Dim FoundUser As Ultimus.OC.User
   Dim ArrayList As System.Collections.ArrayList
   InputShortName = "myshortname"
   InputFullName = ""
   InputUserId = 0
   UltOrgChart.FindUser(InputShortName, InputFullName, InputUserId, FoundUser)
       If FoundUser.GetOCSubOrdinates(ArrayList) Then
          Dim obj As System.Object
          Dim type As System.Type
          If Not ArrayList Is Nothing Then
             For i As Integer = 0 To ArrayList.Count-1
                obj = ArrayList(i)
                type = obj.GetType()
                Select Case type.FullName
                   Case "Ultimus.OC.User"
                      Dim user As Ultimus.OC.User
                      user = obj
                      If (user.strUserName Is Nothing) Then
                         System.Console.WriteLine("Subordinate is job" + _
                         "Function Group" + user.strJobFunction)
                      Else
                         System.Console.WriteLine("Subordinate is User" + _
                         user.strUserName)
                      End If

                   Case "Ultimus.OC.Department"
                      Dim Dept As Ultimus.OC.Department
                      Dept = obj
                      System.Console.WriteLine("Subordinate is " + _
                      "Department" + Dept.strDepartmentName)
                End Select
             Next i
          End If
       End If
```

GetSubordinates

Method: GetSubordinates	

Method description: This method acquires the subordinates of the user.
C# method call:
`public bool GetSubordinates(out Ultimus.OC.User[] SubordinatesArr)`
VB.NET method call:
`Public Function GetSubordinates(ByRef SubordinatesArr() As Ultimus.OC.User) As Boolean`	

Output	
`SubordinatesArr`	**Output Type:** Ultimus.OC.User **Description:** `SubordinatesArr` returns the subordinates of the user.
Return Conditions	
`GetSubordinates` returns `TRUE` if `GetSubordinates` was able to execute. `GetSubordinates` returns `FALSE` if `GetSubordinates` could not execute.	

GetSupervisor

Method: GetSupervisor	

Method description: This method acquires the supervisor of the user.
C# method call:
`public bool GetSupervisor(out string strSupervisor)`
VB.NET method call:
`Public Function GetSupervisor(ByRef strSupervisor As String) As Boolean`	

Output	
`strSupervisor`	**Output Type:** System.String **Description:** `strSupervisor` returns the supervisor of the user.
Return Conditions	
`GetSupervisor` returns `TRUE` if `GetSupervisor` was able to execute. `GetSupervisor` returns `FALSE` if `GetSupervisor` could not execute.	

GetSupervisorInDepartment

Method: GetSupervisorInDepartment	

Method description: This method acquires the supervisor in a specified department.
 	
C# method call:
`public bool GetSupervisorInDepartment(string strDepartment, out string strSupervisor)`
 	
VB.NET method call:
`Public Function GetSupervisorInDepartment(ByVal strDepartment As String, ByRef strSupervisor As String) As Boolean`	

Input	
`strDepartment`	**Input Type:** System.String **Description:** `strDepartment` represents the name of the department whose supervisor is to be retrieved.
Output	
`strSupervisor`	**Output Type:** System.String **Description:** `strSupervisor` represents the name of the supervisor in the specified department.
Return Conditions	
`GetSupervisorInDepartment` returns `TRUE` if `GetSupervisorInDepartment` was able to execute successfully. `GetSupervisorInDepartment` returns `FALSE` if `GetSupervisorInDepartment` could not execute.	

GetSupervisorOfJobFunction

Method: GetSupervisorOfJobFunction	

Method description: This method acquires the supervisor in a specified job function.
 	
C# method call:
`public bool GetSupervisorOfJobFunction(string strJobFunction, out string strSupervisor)`
 	
VB.NET method call:
`Public Function GetSupervisorOfJobFunction(ByVal strJobFunction As String, ByRef strSupervisor As String) As Boolean`	

Input	
`strJobFunction`	**Input Type:** System.String **Description:** `strJobFunction` represents the job function of the supervisor being sought.
Output	
`strSupervisor`	**Output Type:** System.String **Description:** `strSupervisor` represents the name of the supervisor for the specified job function.
Return Conditions	
`GetSupervisorOfJobFunction` returns `TRUE` if `GetSupervisorOfJobFunction` was able to execute successfully. `GetSupervisorOfJobFunction` returns `FALSE` if `GetSupervisorOfJobFunction` could not execute.	

GetUserDepartments

Method: GetUserDepartments	

Method description: This method retrieves all the departments to which a user belongs.
C# method call:
`public bool GetUserDepartments(out Ultimus.OC.Department[] DepartmentList)`
VB.NET method call:
`Public Function GetUserDepartments(ByRef DepartmentList() As Ultimus.OC.Department) As Boolean`	

Output	
`DepartmentList`	**Output Type:** Array of `Ultimus.OC.Department` **Description:** `DepartmentList` represents the departments to which the user belongs.
Return Conditions	
`GetUserDepartments` returns `TRUE` if `GetUserDepartments` was able to execute successfully. `GetUserDepartments` returns `FALSE` if `GetUserDepartments` could not execute.	

GetUserGroups

Method: GetUserGroups	

Method description: This method retrieves the names of all group to which a specified user belongs.
C# method call:
`public bool GetUserGroups(out string[] Groups)`
VB.NET method call:
`Public Function GetUserGroups(ByRef Groups() As String) As Boolean`	

Output	
`Groups`	**Output Type:** System.String **Description:** `Groups` returns a string array of groups to which the user belongs.
Return Conditions	
`GetUserGroups` returns `TRUE` if `GetUserGroups` was able to execute. `GetUserGroups` returns `FALSE` if `GetUserGroups` could not execute.	

GetUserGroups

Method: GetUserGroups	

Method description: This method retrieves the names of all groups to which a user belongs.
C# method call:
`public bool GetUserGroups(out Ultimus.OC.Group[] GroupList)`
VB.NET method call:
`Public Function GetUserGroups(ByRef GroupList() As Ultimus.OC.Group) As Boolean`	

Output	
`GroupList`	**Output Type:** Array of `Ultimus.OC.Group` **Description:** `GroupList` represents the groups to which the user belongs.
Return Conditions	
`GetUserGroups` returns `TRUE` if `GetUserGroups` was able to execute successfully. `GetUserGroups` returns `FALSE` if `GetUserGroups` could not execute.	

GetUserJFGs

Method: GetUserJFGs	

Method description: This method retrieves all the job functions in which a user has.
C# method call:
`public bool GetUserJFGs(out Ultimus.OC.Group[] JFGList)`
VB.NET method call:
`Public Function GetUserJFGs(ByRef JFGList() As Ultimus.OC.Group) As Boolean`	

Output	
`JFGList`	**Output Type:** Array of `Ultimus.OC.Group` **Description:** `JFGList` represents the list of user job functions.
Return Conditions	
`GetUserJFGs` returns `TRUE` if `GetUserJFGs` was able to execute successfully. `GetUserJFGs` returns `FALSE` if `GetUserJFGs` could not execute.	

GetUserPrefs

Method: GetUserPrefs	

Method description: This method retrieves preferences for the user.
 	
C# method call:
`public bool GetUserPrefs(out Ultimus.ClientServices.UserPreferences UserPrefs)`
 	
Method call:
`Public Function GetUserPrefs(ByRef UserPrefs As Ultimus.ClientServices.UserPreferences) As Boolean`	

Output	
`userPrefs`	**Output Type:** instance of `Ultimus.ClientServices.UserPreferences` **Description:** `userPrefs` returns preferences for the user.
Return Conditions	
`GetUserPrefs` returns `TRUE` if `GetUserPrefs` was able to execute. `GetUserPrefs` returns `FALSE` if `GetUserPrefs` could not execute.	

GetUserRights

Method: GetUserRights	

Method description: This method acquires user rights for the user.
 	
C# method call:
`public void GetUserRights(out Ultimus.OC.UserRights rights)`
 	
VB.NET method call:
`Public Sub GetUserRights(ByRef rights As Ultimus.OC.UserRights)`	

Output	
`lRights`	**Output Type:** `Ultimus.OC.UserRights` **Description:** `lRights` corresponds to the access rights of a user within Ultimus Adaptive BPM Suite. Returned values correspond to values in the *ENUMs for `OC.UserRights` class* (outlined on page 115).

Example

```
'This code snippet is written in VB.NET
Dim UltOrgChart As New Ultimus.OC.OrgChart
Dim oUser As Ultimus.OC.User
Dim rights As Ultimus.OC.UserRights

UltOrgChart.FindUser("ShortName", "FullName", "UserID", oUser)
oUser.GetUserRights(right)
If (rights And Ultimus.OC.UserRights.Reports) = Ultimus.OC.UserRights.Reports Then
   MessageBox.Show("You have Access to views configuration")
Else
   MessageBox.Show("Access is denied")
End If
```

GetViewList

Method: GetViewList	

Method description: This method acquires a list of view names for the user.
C# method call:
`public bool GetViewList(out string[] listOfViews)`
VB.NET method call:
`Public Function GetViewList(ByRef listOfViews() As String) As Boolean`	

Output	
`listOfViews`	**Output Type:** System.String **Description:** `listOfViews` returns a list of view names for the user.
Return Conditions	
`GetViewList` returns `TRUE` if `GetViewList` was able to execute. `GetViewList` returns `FALSE` if `GetViewList` could not execute.	

IsJFG

Method: IsJFG

Method description: This method returns `TRUE` if a specified object is a Job Function Group.
C# method call:
`public bool IsJFG()`
VB.NET method call:
`Public Function IsJFG() As Boolean`

Return Conditions
`IsJFG` returns `TRUE` if the specified object is a Job Function Group. `IsJFG` returns `FALSE` if the specified object is not a Job Function Group.

IsMemberOfGroup

Method: IsMemberOfGroup	

Method description: This method returns TRUE if the user is a member of the group specified.
C# method call:
`public bool IsMemberOfGroup(string strGroupName)`
VB.NET method call:
`Public Function IsMemberOfGroup(ByVal strGroupName As String) As Boolean`	

Input	
`strGroupName`	**Input Type:** System.String **Description:** `strGroupName` represents the group name used to query whether a user is a member.
Return Conditions	
`IsMemberOfGroup` returns `TRUE` if the specified user is a member of the specific group. `IsMemberOfGroup` returns `FALSE` if the specified user is not a member of the specific group.	

SetAssociates

Method: SetAssociates	

Method description: This method sets a specified user's associates.
C# method call:
`public bool SetAssociates(Ultimus.ClientServices.UserAssociate[] listUserAssociates)`
VB.NET method call:
`Public Function SetAssociates(ByVal listUserAssociates() As Ultimus.ClientServices.UserAssociate) As Boolean`	

Input	
`listUserAssociates`	**Input Type:** An object array of `Ultimus.ClientServices.UserAssociate` class **Description:** `listUserAssociates` represents the user's associates.
Return Conditions	
`SetAssociates` returns `TRUE` if `SetAssociates` is executed successfully. `SetAssociates` returns `FALSE` if `SetAssociates` is unable to execute successfully.	

SetUserPrefs

Method: SetUserPrefs	

Method description: This method sets the user preferences.
 	
C# method call:
`public bool SetUserPrefs(Ultimus.ClientServices.UserPreferences UserPref)`
 	
VB.NET method call:
`Public Function SetUserPrefs(ByVal UserPref As Ultimus.ClientServices.UserPreferences) As Boolean`	

Input	
`oUserPref`	**Input Type:** instance of `Ultimus.ClientServices.UserPreferences` class **Description:** `oUserPref` sets the user's preferences.
Return Conditions	
`SetUserPrefs` returns `TRUE` if `SetUserPrefs` was able to execute. `SetUserPrefs` returns `FALSE` if `SetUserPrefs` could not execute.	
Example	

```
'This code snippet is written in VB.NET
Dim UltOrgChart As New Ultimus.OC.OrgChart
Dim oUser As Ultimus.OC.User
Dim oUserPref As New Ultimus.ClientServices.UserPreferences

UltOrgChart.FindUser("myname", "", "", oUser)
oUser.GetUserPrefs(oUserPref)
oUserPref.EmailAddress = "myname@ultimus.com"
oUser.SetUserPrefs(oUserPref)
```

The Ultimus EIK interface: classes within the `Configuration` namespace

`Configuration` namespace contains three classes (shown in Figure 6). Each class contains methods and properties.

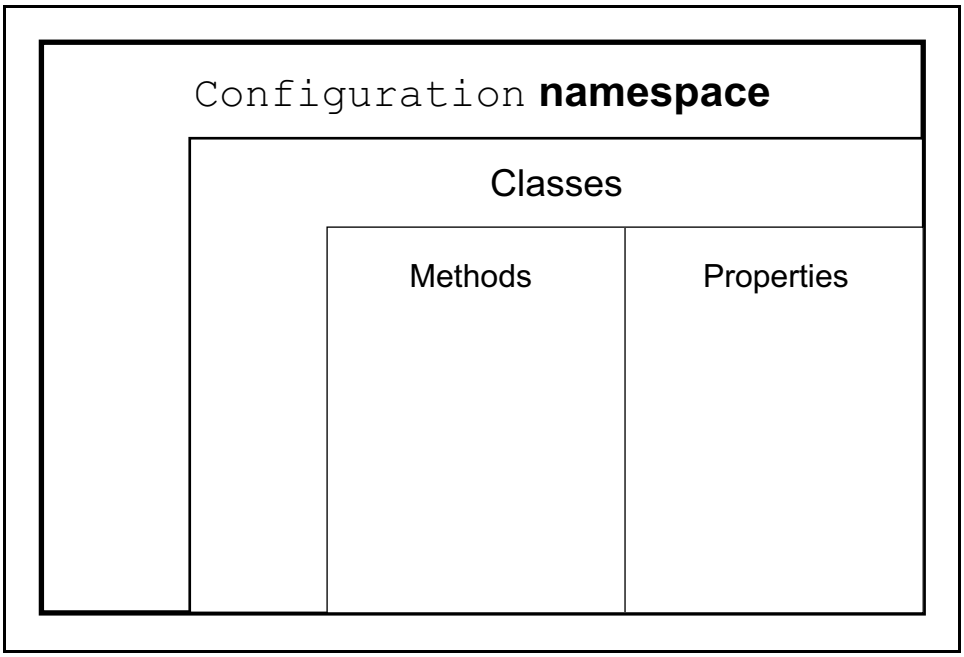

Figure 6. The Ultimus EIK architectural relationship

Outlined in the tables below are descriptions of all status codes (enumerations), properties and methods within each class of the `Configuration` namespace. This section contains:

- *Enumerations (ENUMs)*
- *Properties in the `CommunityInfo` class*
- *Properties in the `LicenseInformation` class*
- *Methods in the `Licensing` class*

Enumerations (ENUMs)

The following table outlines status codes pertaining to user rights. These are defined by the enumerations (ENUMs) in `Configuration.LicenseTypes` class of the `ultikstructures.dll` file.

Table 19. Enumerations for `Configuration.LicenseTypes` class

Enumeration	Description
`CPU`	The license type is based on maximum number of CPUs
`Transaction`	The license type is based on the number of transactions

Properties in the `CommunityInfo` class

The `CommunityInfo` class describes attributes regarding a community. Outlined below are the properties for the `CommunityInfo` class (in alphabetical order).

Table 20. Properties for the `CommunityInfo` class

Property Name	Property Type and Description
`Community`	**Type:** System.String **Description:** `Community` represents the name of the configured community.
`GroupName`	**Type:** System.String **Description:** `GroupName` represents the name of the group belonging to the community.
`Organization`	**Type:** System.String **Description:** `Organization` represents the organization to which the community belongs.

Properties in the `LicenseInformation` class

The `LicenseInformation` class describes attributes regarding customer licensing. Outlined below are the properties for the `LicenseInformation` class (in alphabetical order).

Table 21. Properties for the `LicenseInformation` class

Property Name	Property Type and Description
`CommunitiesCount`	**Type:** System.Int32 **Description:** `CommunitiesCount` represents the total count of licensed community clients.
`ConcurrentClients`	**Type:** UInteger **Description:** `ConcurrentClients` represents the maximum number of users that can be participating in business processes at any given time.
`LicenseType`	**Type:** instance of `Ultimus.Configuration.LicenseInformation.LicenseTypes` class **Description:** `LicenseType` represents the type of license in use.
`MaxCPUs`	**Type:** UInteger **Description:** `MaxCPUs` represents the maximum number of CPUs that can be present on the computer hosting Ultimus BPM Server.
`MaxTransPerDay`	**Type:** UInteger **Description:** `MaxTransPerDay` represents the maximum number of Ultimus engine tasks that can be completed by Ultimus BPM Server within a day (24 hour period).
`NamedClients`	**Type:** UInteger **Description:** `NamedClients` represents the maximum number of individual process participants for whom Ultimus BPM Server can perform a task.

Methods in the `Licensing` class

The `Licensing` class provides the functions to configure customer licensing. Outlined below are the methods for the `Licensing` class (in alphabetical order):

- *AddNamedClient*
- *AddUserToConcurrentList*
- *GetCommunityClientInfo*
- *GetConcurrentLicenseInfo*

- *GetLicenseInfo*
- *GetNamedClientLicenseInfo*
- *RemoveNamedClient*
- *RemoveUserFromConcurrentList*

AddNamedClient

Method: AddNamedClient	

Method description: This method adds a user to the Named Clients list.
C# method call:
`public bool AddNamedClient(string strUserName, string strOrg, out string strErr)`
VB.NET method call:
`Public Function AddNamedClient(ByVal strUserName As String, ByVal strOrg As String, ByRef strErr As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the name of the user who is to be added in the Named Clients list.
`strOrg`	**Input Type:** System.String **Description:** `strOrg` represents the name of the organization to which the specified user belongs.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to add a named client.
Return Conditions	
`AddNamedClient` returns `TRUE` if `AddNamedClient` is executed successfully. `AddNamedClient` returns `FALSE` if `AddNamedClient` is unable to execute successfully.	

AddUserToConcurrentList

Method: AddUserToConcurrentList	

Method description: This method adds a user to the Concurrent Client list.
 	
C# method call:
`public bool AddUserToConcurrentList(string strUserName, string strOrg, out string strErr)`
 	
VB.NET method call:
`Public Function AddUserToConcurrentList(ByVal strUserName As String, ByVal strOrg As String, ByRef strErr As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the name of the user who is to be added in the Concurrent Client list.
`strOrg`	**Input Type:** System.String **Description:** `strOrg` represents the name of the organization to which the specified user belongs.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to add a user to the Concurrent Client list.
Return Conditions	
`AddUserToConcurrentList` returns `TRUE` if `AddUserToConcurrentList` is executed successfully. `AddUserToConcurrentList` returns `FALSE` if `AddUserToConcurrentList` is unable to execute successfully.	

GetCommunityClientInfo

Method: GetCommunityClientInfo	

Method description: This method retrieves Community Client license information.
 	
C# method call:
`public bool GetCommunityClientInfo(out Ultimus.Configuration.CommunityInfo[] communityInfo, out string strErr)`
 	
VB.NET method call:
`Public Function GetCommunityClientInfo(ByRef communityInfo() As Ultimus.Configuration.CommunityInfo, ByRef strErr As String) As Boolean`	

Output	
`communityInfo`	**Output Type:** instance of `CommunityInfo` class **Description:** `communityInfo` represents an object of `CommunityInfo` class with community client information.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to get the community client information.
Return Conditions	
`GetCommunityClientInfo` returns `TRUE` if `GetCommunityClientInfo` is executed successfully. `GetCommunityClientInfo` returns `FALSE` if `GetCommunityClientInfo` is unable to execute successfully.	

GetConcurrentLicenseInfo

Method: GetConcurrentLicenseInfo	

Method description: This method retrieves Concurrent Client license information.
 	
C# method call:
`public bool GetConcurrentLicenseInfo(out string[] strConcurrentUsers, out string strErr)`
 	
VB.NET method call:
`Public Function GetConcurrentLicenseInfo(ByRef strConcurrentUsers() As String, ByRef strErr As String) As Boolean`	

Output	
`strConcurrentUsers`	**Output Type:** System.String **Description:** `strConcurrentUsers` represents a string array of Concurrent Client users.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to get the concurrent user license information.
Return Conditions	
`GetConcurrentLicenseInfo` returns `TRUE` if `GetConcurrentLicenseInfo` is executed successfully. `GetConcurrentLicenseInfo` returns `FALSE` if `GetConcurrentLicenseInfo` is unable to execute successfully.	

GetLicenseInfo

Method: GetLicenseInfo	

Method description: This method retrieves the license information.
 	
C# method call:
`public bool GetLicenseInfo(out Ultimus.Configuration.LicenseInformation LicenseInfo, out string strErr)`
 	
VB.NET method call:
`Public Function GetLicenseInfo(ByRef LicenseInfo As Ultimus.Configuration.LicenseInformation, ByRef strErr As String) As Boolean`	

Output	
`LicenseInfo`	**Output Type:** `Ultimus.Configuration.LicenseInformation` **Description:** `LicenseInfo` represents the license information.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to get the license information.
Return Conditions	
`GetLicenseInfo` returns `TRUE` if `GetLicenseInfo` is executed successfully. `GetLicenseInfo` returns `FALSE` if `GetLicenseInfo` is unable to execute successfully.	

GetNamedClientLicenseInfo

Method: GetNamedClientLicenseInfo	

Method description: This method retrieves the list of Named Clients.
 	
C# method call:
`public bool GetNamedClientLicenseInfo(out string[] strNamedUsers, out string strErr)`
 	
VB.NET method call:
`Public Function GetNamedClientLicenseInfo(ByRef strNamedUsers() As String, ByRef strErr As String) As Boolean`	

Output	
`strNamedUsers`	**Output Type:** System.String **Description:** `strNamedUsers` represents a string array of Named Client users.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to get the Named Clients license information.
Return Conditions	
`GetNamedClientLicenseInfo` returns `TRUE` if `GetNamedClientLicenseInfo` is executed successfully. `GetNamedClientLicenseInfo` returns `FALSE` if `GetNamedClientLicenseInfo` is unable to execute successfully.	

RemoveNamedClient

Method: RemoveNamedClient	

Method description: This method removes a specified user from the Named Clients list.
 	
C# method call:
`public bool RemoveNamedClient(string strUserName, string strOrg, out string strErr)`
 	
VB.NET method call:
`Public Function RemoveNamedClient(ByVal strUserName As String, ByVal strOrg As String, ByRef strErr As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the name of the user who is to be removed from the Named Client list.
`strOrg`	**Input Type:** System.String **Description:** `strOrg` represents the name of the organization to which the specified user belongs.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to remove the user from the Named Clients list.
Return Conditions	
`RemoveNamedClient` returns `TRUE` if `RemoveNamedClient` is executed successfully. `RemoveNamedClient` returns `FALSE` if `RemoveNamedClient` is unable to execute successfully.	

RemoveUserFromConcurrentList

Method: RemoveUserFromConcurrentList	

Method description: This method removes a specified user from the Concurrent Clients list.
 	
C# method call:
`public bool RemoveUserFromConcurrentList(string strUserName, string strOrg, out string strErr)`
 	
VB.NET method call:
`Public Function RemoveUserFromConcurrentList(ByVal strUserName As String, ByVal strOrg As String, ByRef strErr As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the name of the user who is to be removed from the Concurrent Clients list.
`strOrg`	**Input Type:** System.String **Description:** `strOrg` represents the name of the organization to which the specified user belongs.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to remove the user from the Concurrent Clients list.
Return Conditions	
`RemoveUserFromConcurrentList` returns `TRUE` if `RemoveUserFromConcurrentList` is executed successfully. `RemoveUserFromConcurrentList` returns `FALSE` if `RemoveUserFromConcurrentList` is unable to execute successfully.	

Using the Ultimus EIK with Ultimus-context information

The Ultimus EIK can be used to retrieve Ultimus-context information. When .NET code is being called from business rules or Ultimus Forms, the .NET code running under incidents and tasks is given a context in order to manipulate those incidents or tasks. This is done only if the .NET code has defined XML schema elements.

The context objects for the incident and the task are the same as those provided by the Ultimus EIK. Two public member variables must be added into the .NET class using the names `ContextIncident` and `ContextTask`. These are described below.

```
namespace TestEikN

{
    public class EikContextTest
    {
        public Ultimus.WFServer.Task ContextTask;
        public Ultimus.WFServer.Incident ContextIncident;
}
}
}
```

In order for the .NET code to compile with these variables, the reference to the `UltEIK.dll` file must be added to the `Global.ref` file. To do this, follow these steps:

1. Open the `Global.ref` using Notepad. The `Global.ref` file is located within the `Resources` directory of the Ultimus Adaptive BPM Suite installation directory.
2. Within the `Global.ref` file, specify the path to the `UltEIK.dll` file (which is located in the Ultimus Adaptive BPM Suite installation directory).

When the .NET code is called, Ultimus BPM Server initializes the `ContextTask` and `ContextIncident` variables to the in-memory task and to the incident to which the code is being called.

For example, when the .NET code is called from a business rule, various properties of the task may be accessed, including the variables within the task and all the methods available in the Ultimus EIK for the task object. These methods directly manipulate the task in memory at Ultimus BPM Server. However, the incident or the task variable can be null in certain cases. These are two examples of when the incident or task variable may be null:

- At the Begin step the context of the incident or task is not available because the incident has not yet been created.
- At an “Activate” condition of any step, the context of the task is not available because the task is created after the execution of the activate condition.

The tables below (Tables 22 through 25) outline each ESI method occurrence and how it maps to a property or method.

In these tables, the letter "Y" represents the ESI method occurrence is supported for that property or method. The letter "X" represents the event condition is not supported for that property or method; if called, an error message `This functionality is not allowed in this context` displays.

The classes outlined in these tables are from the `WFServer` namespace:

- *`Incident` class*
- *`Incident.Status` class*
- *`Incident.Status.StepStatus` class*
- *`Task` class*

`Incident` class

The following table outlines which event conditions are supported under the properties and methods of the `Incident` class.

Table 22. Supported event conditions for properties and methods in the `Incident` class

	Activate	Complete	Return	Resubmit	Late	Recipient	Form
Properties							
`nIncidentNo`	**Y**	**Y**	**Y**	**Y**	**Y**	**X**	**Y**
`nVersion`	**Y**	**Y**	**Y**	**Y**	**Y**		**Y**
`strIncident Owner`	**Y**	**Y**	**Y**	**Y**	**Y**		**Y**
`strIncident Summary`	**Y**	**Y**	**Y**	**Y**	**Y**		**Y**
`strProcessName`	**Y**	**Y**	**Y**	**Y**	**Y**		**Y**

Table 22. Supported event conditions for properties and methods in the `Incident` class *(Continued)*

	Activate	Complete	Return	Resubmit	Late	Recipient	Form
Methods							
AbortIncident	X	Y	Y	Y	Y	X	Y
AbortStep	Y	Y	Y	Y	Y		X
AddMemoEntry	Y	Y	Y	Y	Y		Y
AppendTo RepeatedNode	Y	Y	Y	Y	Y		Y
DeleteIncident	X	X	X	X	X		Y
DirectActivate Step	X	X	X	X	X		X
DirectActivate StepEx	X	Y	Y	Y	Y		Y
GetIncident Status	Y	Y	Y	Y	Y		Y
GetIncidentXML	Y	Y	Y	Y	Y		Y
GetMapVersion	Y	Y	Y	Y	Y		Y
GetMemo	Y	Y	Y	Y	Y		Y
GetNodeValue	Y	Y	Y	Y	Y		Y
GetProcess Schema	Y	Y	Y	Y	Y		Y
GetRepeatedNodeS ize	Y	Y	Y	Y	Y		Y
GetStepStatus	Y	Y	Y	Y	Y		Y
GetVariable ValueAtIndex	Y	Y	Y	Y	Y		Y
GetVariable Values	Y	Y	Y	Y	Y		Y
InsertInto RepeatedNode	Y	Y	Y	Y	Y		Y
LoadIncident	X	X	X	X	X		Y
RemoveRepeated NodeIndex	Y	Y	Y	Y	Y		Y

Table 22. Supported event conditions for properties and methods in the `Incident` class *(Continued)*

	Activate	Complete	Return	Resubmit	Late	Recipient	Form
`SaveVariables`	Y	Y	Y	Y	Y	X	Y
`SetIncidentXML`	Y	Y	Y	Y	Y		Y
`SetNodeValue`	Y	Y	Y	Y	Y		Y

`Incident.Status` class

The following table outlines which ESI method occurrences are supported under the properties and methods of the `Incident.Status` class.

Table 23. Supported ESI method occurrences for properties and methods in the `Incident.Status` class

	Activate	Complete	Return	Resubmit	Late	Recipient	Form
Properties							
`dCompleteTime`	X	X	X	X	X	X	X
`dLateTime`	Y	Y	Y	Y	Y		Y
`dStartTime`	Y	Y	Y	Y	Y		Y
`nIncidentStatus`	Y	Y	Y	Y	Y		Y
`TaskStatuses`	Y	Y	Y	Y	Y		Y
Methods							
`GetGraphical Status`	Y	X	X	X	X	X	Y

`Incident.Status.StepStatus` class

The following table outlines which ESI method occurrences are supported under the properties and methods of the `Incident.Status.StepStatus` class.

Table 24. Supported ESI method occurrences for properties and methods in the `Incident.Status.StepStatus` class

	Activate	Complete	Return	Resubmit	Late	Recipient	Form
Properties							
`dtEndTime`	Y	Y	Y	Y	Y	X	Y
`dtStartTime`	Y	Y	Y	Y	Y		Y
`nStep Properties`	Y	Y	Y	Y	Y		Y
`nStepStatus`	Y	Y	Y	Y	Y		Y
`nStepSubStatus`	Y	Y	Y	Y	Y		Y
`nStepType`	Y	Y	Y	Y	Y		Y
`strStepName`	Y	Y	Y	Y	Y	X	Y
`strStep Recipient`	Y	Y	Y	Y	Y		Y

`Task` class

The following table outlines which ESI method occurrences are supported under the properties and methods of the `Task` class.

Table 25. Supported ESI method occurrences for properties and methods in the `Task` class

	Activate	Complete	Return	Resubmit	Late	Recipient	Form
Properties							
`dCost`	X	Y	X	Y	X	X	X
`dDelayTime`		Y	Y	Y	Y		Y
`dEndTime`		X	X	X	X		X
`dOverDueTime`		Y	Y	Y	Y		Y
`dQueueEndTime`		Y	Y	Y	Y		Y

Table 25. Supported ESI method occurrences for properties and methods in the `Task` class *(Continued)*

	Activate	Complete	Return	Resubmit	Late	Recipient	Form
dQueueStartTime	X	Y	Y	Y	Y	X	Y
dStartTime		Y	Y	Y	Y		Y
dUrgentTime		Y	Y	Y	Y		Y
nIncidentNo		Y	Y	Y	Y		Y
nPriority		Y	Y	Y	Y		Y
nProcessVersion		Y	Y	Y	Y		Y
nRecipientType		Y	Y	Y	Y		Y
nStepProperties		Y	Y	Y	Y		Y
nStepType		Y	Y	Y	Y		Y
nTaskRate		Y	Y	Y	Y		Y
nTaskStatus		Y	Y	Y	Y		Y
nTaskSubStatus		Y	Y	Y	Y		Y
nTaskTime		Y	X	Y	X		X
strAssignedTo User		Y	Y	Y	Y		Y
strAssignedTo UserFullName		Y	Y	Y	Y		X
strFormUrl		Y	Y	Y	Y		Y
strGroupName		Y	Y	Y	Y		Y
strHelpUrl		Y	Y	Y	Y		Y
strProcessName		Y	Y	Y	Y		Y
strRecipient		Y	Y	Y	Y		Y
strRecipient FullName		X	X	X	X		X
strStepId		Y	Y	Y	Y		Y
strStepOrg		Y	Y	Y	Y		Y
strStepName		Y	Y	Y	Y		Y

Table 25. Supported ESI method occurrences for properties and methods in the `Task` class *(Continued)*

	Activate	Complete	Return	Resubmit	Late	Recipient	Form
strSummary	X	Y	Y	Y	Y	X	Y
strTaskId		Y	Y	Y	Y		Y
strUser		Y	Y	Y	Y		Y
strUserFullName		X	X	X	X		X
Methods							
AbortStep	X	X	X	X	Y	X	Y
AddMemoEntry		Y	Y	Y	Y		Y
AppendTo RepeatedNode		X	Y	Y	Y		Y
AssignQueueTask		X	X	X	Y		Y
AssignTask		X	X	X	Y		Y
DeleteTask		X	X	X	X		Y
ExtractFormURL		Y	Y	Y	Y		Y
GetMapVersion		Y	Y	Y	Y		Y
GetMemo		Y	Y	Y	Y		Y
GetNodeValue		Y	Y	Y	Y		Y
GetProcessHelp URL		Y	Y	Y	Y		Y
GetRepeatedNode Size		Y	Y	Y	Y		Y
GetStepHelpURL		Y	Y	Y	Y		Y
GetStepSchema		Y	Y	Y	Y		Y
GetTaskXML		Y	Y	Y	Y		Y
InitializeFrom InitiateTaskId		Y	Y	Y	Y		Y
InitializeFrom NextQueueTask		X	X	X	X		Y
InitializeFrom TaskId		Y	Y	Y	Y		Y

Table 25. Supported ESI method occurrences for properties and methods in the `Task` class *(Continued)*

	Activate	Complete	Return	Resubmit	Late	Recipient	Form
`InsertInto RepeatedNode`	X	X	Y	Y	Y	X	Y
`PutTaskInQueue`		X	X	X	Y		Y
`Refresh`		X	X	X	Y		Y
`RemoveRepeated NodeIndex`		Y	Y	Y	Y		Y
`Return`		X	X	X	Y		Y
`SaveTemplate`		X	X	X	X		X
`SaveXML`		X	X	X	X		Y
`Send`		X	X	X	Y		Y
`SendFrom`		X	X	X	Y		Y
`SetNodeValue`		Y	Y	Y	Y		Y
`SetTaskXML`		Y	Y	Y	Y		Y
`TakeBack`		X	X	X	Y		Y

2

Events Subscription Interface

This section describes how event systems function in general, as well as how an event subscription interface functions within the Ultimus EIK.

Events-based publish/subscribe systems provide functionality where applications can subscribe to events that occur on a target system, and then those applications are proactively notified by the target system when particular events occur within that server. The Ultimus Events Subscription Interface (ESI) includes a COM+ Loosely Coupled Events (LCE) interface, where information on published events is stored within the COM+ catalog store. Subscribers can query this store and create a subscription by selecting the events that they would like to have information about when an event occurs. Then when an event occurs, the event system would check the store to find the interested subscribers, create a new object of each interested class, and call to the methods on that object.

An example of an Ultimus ESI event would be when an incident is aborted in Ultimus Process Administrator. When this happens, the Ultimus ESI calls the `UltimusEvents.IUltimusPublisher.IncidentAborted` method. As part of this method execution, Ultimus-specific information is passed to the method. In this example, process name and incident number are passed to the `IncidentAborted` method. So based upon this notification, the user will not only come to know what has happened in the Ultimus engine but will also receive the event details as parameters.

Such information can become very useful. For example, the Ultimus ESI could be useful in a scenario where an auditing application is required to track key events that occur in an Ultimus BPM Server (such as when users initiate processes, complete their tasks, or other events). At run-time, when an event is provided to the auditing application, it could write this information to a third-party database.

In overview, to construct the Ultimus EIK subscription interface requires three procedures:

1. *Creating an event class*
2. *Creating a custom subscriber class*
3. *Installing the custom subscription class*

Each of these procedures is discussed below.

Creating an event class

Through the Ultimus EIK, Ultimus has already created an event class. As part of this class, the Ultimus EIK exposes many event-driven Ultimus engine methods. These Ultimus engine methods are (in alphabetical order):

- *CompletedTaskDeleted*
- *IncidentAborted*
- *IncidentCompleted*
- *IncidentInitiated*
- *QueueTaskActivated*
- *StepAborted*
- *TaskActivated*
- *TaskAssigned*
- *TaskCompleted*
- *TaskConferred*
- *TaskDelayed*
- *TaskDeletedOnMinResponseComplete*
- *TaskLate*
- *TaskResubmitted*
- *TaskReturned*
- *TasksPerDayThresholdReached*
- *TaskSubmitFailed*

CompletedTaskDeleted

Method: CompletedTaskDeleted
Method description: This method is called when a completed task is deleted.
VB.NET method call:
`Public Sub CompletedTaskDeleted(ByVal strProcessName As String, ByVal nIncident As Integer, ByVal strTaskId As String)`

IncidentAborted

Method: IncidentAborted
Method description: This method is called when an incident is aborted. **VB.NET method call:** `Public Sub IncidentAborted(ByVal strProcessName As String, ByVal nIncident As Integer, ByVal strReason As String)`

IncidentCompleted

Method: IncidentCompleted
Method description: This method is called when an incident is completed. **VB.NET method call:** `Public Sub IncidentCompleted(ByVal strProcessName As String, ByVal nIncident As Integer)`

IncidentInitiated

Method: IncidentInitiated
Method description: This method is called when an incident is initiated. **VB.NET method call:** `Public Sub IncidentInitiated(ByVal strProcessName As String, ByVal nIncident As Integer)`

QueueTaskActivated

Method: QueueTaskActivated
Method description: This method is called when a Queue task is activated. **VB.NET method call:** `Public Overridable Sub QueueTaskActivated(ByVal strProcessName As String, ByVal nIncident As Integer, ByVal strTaskId As String)`

StepAborted

Method: StepAborted
Method description: This method is called when a task is aborted. **VB.NET method call:** `Public Sub StepAborted(ByVal strProcessName As String, ByVal nIncident As Integer, ByVal nStepType As Integer, ByVal strStepId As String, ByVal strStepLabel As String)`

TaskActivated

Method: TaskActivated
Method description: The method is called under the following circumstances: • `TaskActivated` is called when an Ultimus step is activated. • For group steps, `TaskActivated` is called for each member of the group to whom the task is activated. • `TaskActivated` is called when a recipient of a task is determined via an Ultimus Director business rule. • `TaskActivated` is called when an Ultimus client user retrieves a task from a queue. • `TaskActivated` is called when the Ultimus BPM engine automatically assigns a queue task to a user. • For sequential group tasks, `TaskActivated` is called only when the task is activated to the first member of the group. **VB.NET method call:** `Public Sub TaskActivated(ByVal strProcessName As String, ByVal nIncident As Integer, ByVal nStepType As Integer, ByVal strTaskId As String)`

TaskAssigned

Method: TaskAssigned
Method description: This method is called when a task is assigned. **VB.NET method call:** `Public Sub TaskAssigned(ByVal strProcessName As String, ByVal nIncident As Integer, ByVal strTaskId As String, ByVal strAssignedUser As String)`

TaskCompleted

Method: TaskCompleted
Method description: This method is called when a task is completed. **VB.NET method call:** `Public Sub TaskCompleted(ByVal strProcessName As String, ByVal nIncident As Integer, ByVal nStepType As Integer, ByVal strTaskId As String)`

TaskConferred

Method: TaskConferred
Method description: This method is called when a task is conferred. **VB.NET method call:** `Public Sub TaskConferred(ByVal strProcessName As String, ByVal nIncident As Integer, ByVal strTaskId As String, ByVal strUser As String)`

TaskDelayed

Method: TaskDelayed
Method description: This method is called when a task is delayed. **VB.NET method call:** `Public Sub TaskDelayed(ByVal strProcessName As String, ByVal nIncident As Integer, ByVal strTaskId As String)`

TaskDeletedOnMinResponseComplete

Method: TaskDeletedOnMinResponseComplete
Method description: This method is called when the minimum response for a step is reached. For example, suppose there is a group on a step and the group contains five users. Suppose also that we have set a minimum response for this step as `3` (in Ultimus BPM Studio). When this step is activated with all the group members and three of them complete the step, the tasks for the other two users are deleted and this event is fired. **VB.NET method call:** `Public Sub TaskDeletedOnMinResponseComplete(ByVal strProcessName As String, ByVal nIncident As Integer, ByVal strTaskId As String)`

TaskLate

Method: TaskLate
Method description: This method is called when a task becomes late/urgent. **VB.NET method call:** `Public Sub TaskLate(ByVal strProcessName As String, ByVal nIncident As Integer, ByVal strTaskId As String)`

TaskResubmitted

Method: TaskResubmitted
Method description: This method is called when a task is resubmitted. **VB.NET method call:** `Public Sub TaskResubmitted(ByVal strProcessName As String, ByVal nIncident As Integer, ByVal strTaskId As String)`

TaskReturned

Method: TaskReturned
Method description: This method is called when a task is returned. **VB.NET method call:** `Public Sub TaskReturned(ByVal strProcessName As String, ByVal nIncident As Integer, ByVal nStepType As Integer, ByVal strTaskId As String)`

TasksPerDayThresholdReached

Method: TasksPerDayThresholdReached
Method description: This method is called when the allowed number of tasks per day has been reached (such as when an Ultimus license expires). This works only with Transaction-based licenses. **VB.NET method call:** `Public Sub TasksPerDayThresholdReached(ByVal lTasksPerDayLimit As Long, ByVal lThreshhold As Long)`

TaskSubmitFailed

Method: TaskSubmitFailed
Method description: This method is called when a task submission fails. **VB.NET method call:** `Public Sub TaskSubmitFailed(ByVal strTaskId As String)`

Creating a custom subscriber class

This section discusses the basics of a custom subscriber class. A subscriber class is provided as an Ultimus Adaptive BPM Suite 8.3 SP2 EIK sample.

At a minimum, a custom subscriber class must implement two other interfaces:

- `System.EnterpriseServices.ServicedComponent` (as declared in the .NET Framework)
- UltimusEvents (as declared in the `UltimusPublisher.dll` file)

An example of how this would look in code would be as follows:

```
Imports System.EnterpriseServices
<Assembly: ApplicationName("UltimusSubscriber")>
Public Class UltimusSubscriberClass
    Inherits System.EnterpriseServices.ServicedComponent
    Implements UltimusEvents.IUltimusPublisher
    //insert custom code here
End Class
```

The end result of creating the custom subscriber class is the generation of an Ultimus subscriber `DLL` file. Ultimus Adaptive BPM Suite 8.3 SP2 also provides a sample subscription interface `DLL` file, called `UltimusSubscriber.dll`. Locate this `DLL` file in the Ultimus Adaptive BPM Suite 8.3 SP2 subscription interface EIK sample `ZIP` directory.

Installing the custom subscription class

After developing the subscription class, the COM+ component must be installed and the subscription configuration performed against the EIK interface `UltimusEvents`.

Installing the subscription class involves three procedures:

1. Add the file `UltimusSubscriber.dll` to a COM+ component.
2. Subscribe the component to the EIK event interface.
3. Enable the EIK subscription interface through Ultimus System Administrator.

These of these procedures is outlined below.

To add `UltimusSubscriber.dll` to a COM+ component, follow these steps:

1. Go to **Start»Control Panel»Administrative Tools»Component Services**. Expand the **Component Services** node then select **Computers»My Computer»COM+ Applications**.
2. Right-click on COM+ Applications, then select **New»Application** (as shown in Figure 7).

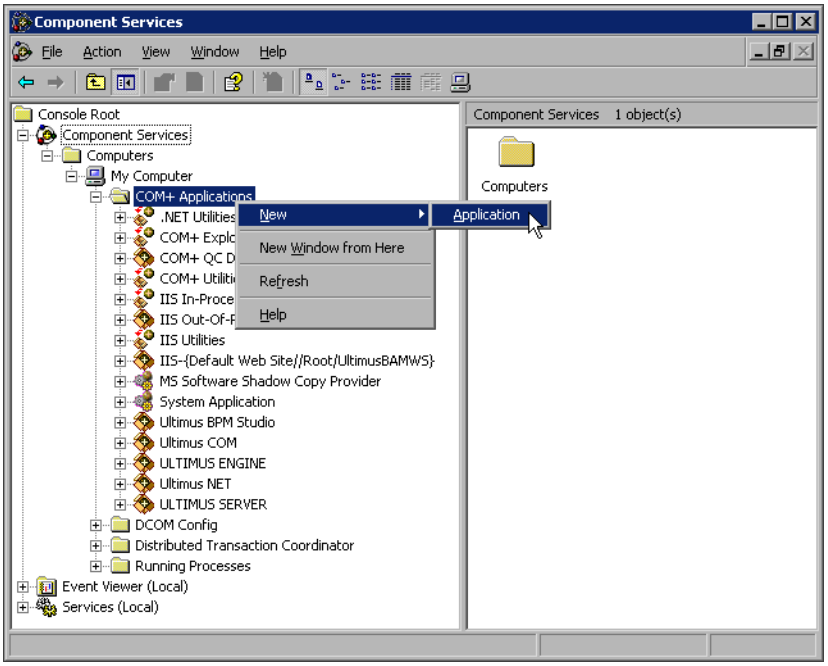

Figure 7. Creating a new COM+ application

3. The **COM+ Application Install Wizard** appears (as shown in Figure 8). Click the **Next** button.

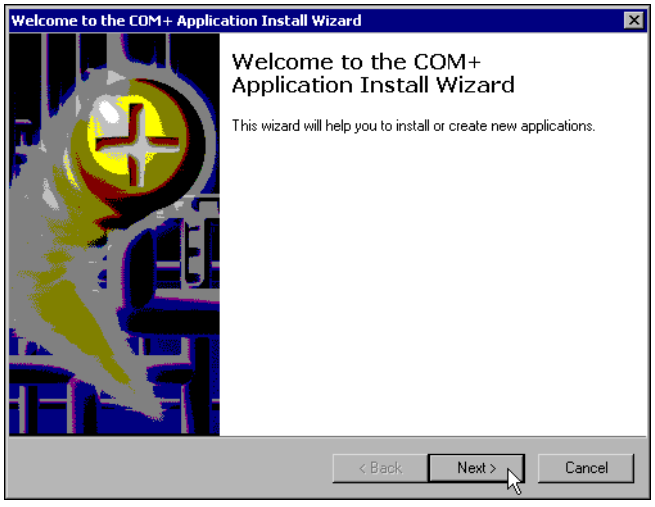

Figure 8. The COM+ Application Install Wizard

4. Click **Create an empty application** button.
5. Enter a name into the **Enter a name for the new application** field. Verify that the **Server application** radio button in the **Activation Type** section is selected. Click the **Next** button.
6. On the **Set Application Identity** page of the wizard, verify the **Interactive user - the current logged on user** radio button is selected in the **Account** section. Click the **Next** button.
7. Click the **Finish** button to complete the **COM+ Application Install Wizard**. The new COM+ application appears as a new node in the **COM+ Applications** section.
8. Expand the COM+ application node that was created.
9. Right-click the **Components** folder, then select **New»Component** (as shown in Figure 9).

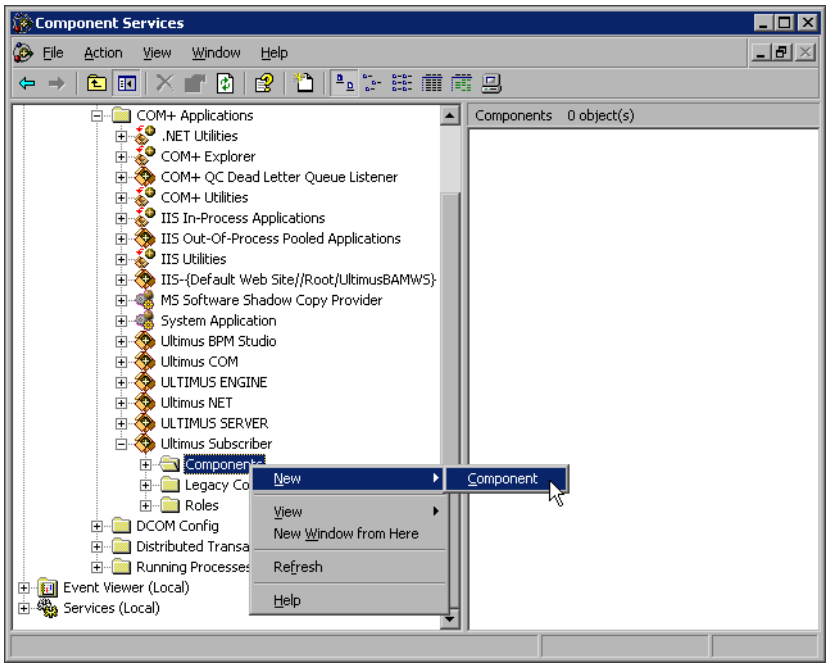

Figure 9. Adding a new component to the COM+ application

10. The **Welcome to the COM+ Component Install Wizard** appears (as shown in Figure 10).

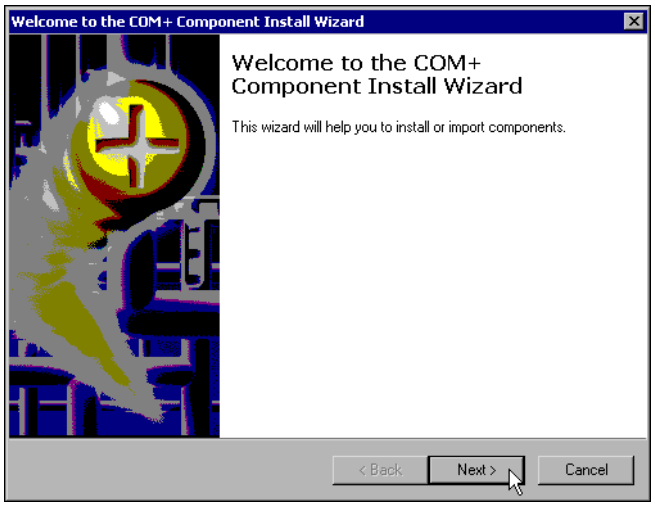

Figure 10. The Welcome to the COM+ Component Install Wizard

11. Click the **Next** button, then click the **Install new component(s)** button.
12. In the **Select files to install** dialog box, browse for the file `UltimusSubscriber.dll`, then select **Open**.
13. In the **Install new components** page of the wizard, the `DLL` file is added, as shown in Figure 11.

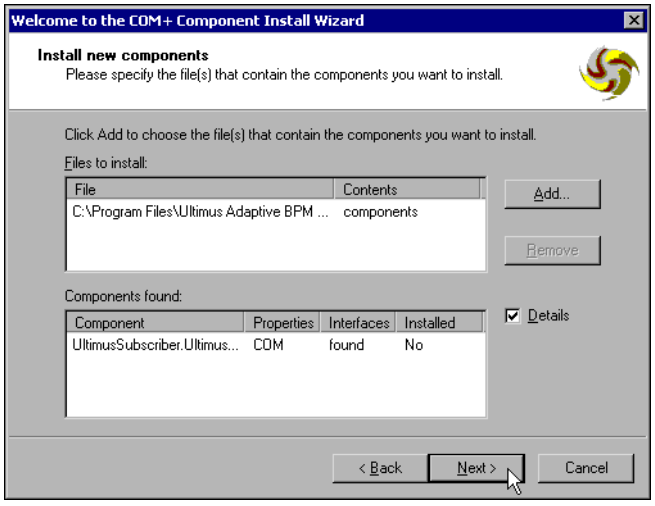

Figure 11. The Install new components page of the COM+ Component Install Wizard

14. Click the **Next** button, then click the **Finish** button in the last page of the wizard. The subscription class is now installed to the COM+ component.

To subscribe the properly added COM+ component to the EIK event interface, follow these steps:

1. Expand the **Components** folder, then expand the **UltimusSubscriber.UltimusSubscriberClass** component node. Right-click on the **Subscriptions** folder, then select **New»Subscription** (as shown in Figure 12).

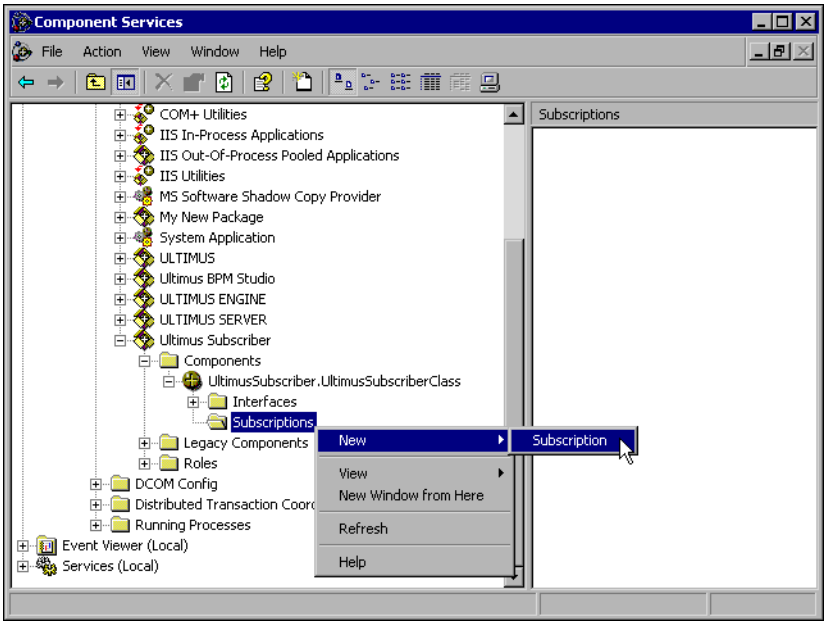

Figure 12. Adding a new subscription to the COM+ component

2. The **Welcome to the COM+ New Subscription Wizard** appears (as shown in Figure 13).

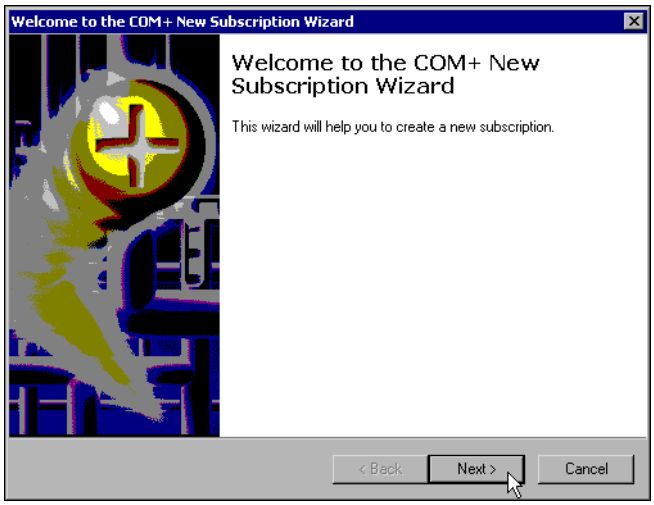

Figure 13. The Welcome to the COM+ New Subscription Wizard

3. Click the **Next** button. In the **Select Subscription Method(s)** page of the wizard, browse to the **IUltimusPublisher** interface, then click the **Next** button (as shown in Figure 14).

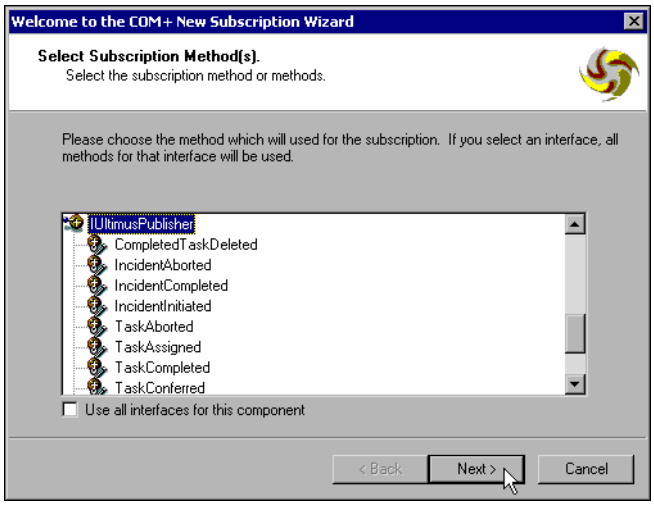

Figure 14. Adding a subscription to IUltimusPublisher subscription class

4. In the **Select Event Class** page of the wizard, select **UltimusEvents.CUltimusPublisher** from the **Prog ID** column. Select the **Next** button.
5. In the **Subscription Options** page of the wizard, place a name of the subscription into the **Enter a name for the new subscription:** field. Select the check box **Enable this subscription immediately**.
6. Click the **Next** button, then click the **Finish** button in the last page of the wizard. A subscription for the COM+ component has been added.

To enable the EIK subscription interface through Ultimus System Administrator, follow these steps:

1. Open Ultimus System Administrator (**Start»Programs»Ultimus Adaptive BPM Suite 8.3»Ultimus System Administrator**).
2. Right-click the Ultimus System Administrator parent node, then select **BPM Server Properties...**.

3. The **BPM Server Properties** dialog box appears. Open the **Advanced** node, then select the **Event Configuration** option, as shown in Figure 15.

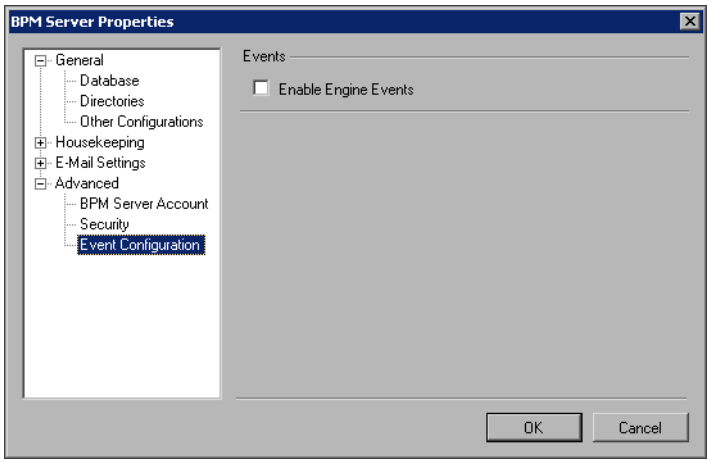

Figure 15. Advanced Properties in Ultimus System Administrator

4. Select the **Enable Engine Events** check box.
5. Click the **OK** button.

The subscription class is now installed, and ready for use.

Tip: Simulation performed in Ultimus BPM Studio Client, when performed on the same computer hosting Ultimus BPM Server, results in the execution of EIK Subscription events. As such, it is not recommended to perform simulation in Ultimus BPM Studio Client on the same computer hosting Ultimus BPM Server.

3

Ultimus BIService Web service

The Ultimus BIService Web service is an interface for business process and task data to third-party applications. The third-party reporting applications can leverage the Ultimus BIService Web service to retrieve business and process level data to produce quick and data-rich dashboards, portals, and reports. Outlined below are descriptions of all methods within the `BIService` namespace (in alphabetical order):

- *GetBPMQuerySchema*
- *GetBPMSchema*
- *GetProcessQuerySchema*
- *GetProcessSchema*
- *GetProcessStepQuerySchema*
- *GetProcessStepSchema*
- *RunBPMQuery*
- *RunProcessQuery*
- *RunProcessStepQuery*

GetBPMQuerySchema

Method: GetBPMQuerySchema	

Method description: This method retrieves the resulting XML schema generated by a specified XQuery.
C# method call:
`public System.Xml.XmlNode GetBPMQuerySchema (string XQuery)`
VB.NET method call:
`Public Function GetBPMQuerySchema (ByVal XQuery As String) As System.Xml.XmlNode`	

Input	
`XQuery`	**Input Type:** System.String **Description:** `XQuery` represents an XQuery that runs on Ultimus XML data.
Return Conditions	
`GetBPMQuerySchema` returns the XML schema of the specified XQuery in an XMLNode format.	

GetBPMSchema

Method: GetBPMSchema

Method description: This method retrieves the standard BIS XML schema.
C# method call:
`public System.Xml.XmlNode GetBPMSchema ()`
VB.NET method call:
`Public Function GetBPMSchema () As System.Xml.XmlNode`

GetProcessQuerySchema

Method: GetProcessQuerySchema	

Method description: This method retrieves the resulting XML schema generated by a specified XQuery executed on a specified business process.
 	
C# method call:
`public System.Xml.XmlNode GetProcessQuerySchema(string process, int version, string XQuery)`
 	
VB.NET method call:
`Public Function GetProcessQuerySchema(ByVal process As String, ByVal version As Integer, ByVal XQuery As String) As System.Xml.XmlNode`	

Input	
`process`	**Input Type:** System.String **Description:** `process` represents the name of the business process on which the Xquery is to run.
`version`	**Input Type:** System.Int32 **Description:** `version` represents the version of the specified business process.
`XQuery`	**Input Type:** System.String **Description:** `XQuery` represents an XQuery that accesses the process XML schema.
Return Conditions	
`GetProcessQuerySchema` returns the XML schema against the specified XQuery.	

GetProcessSchema

Method: GetProcessSchema	

Method description: This method retrieves the process XML schema for a specified business process.
 	
C# method call:
`public System.Xml.XmlNode GetProcessSchema (string process, int version)`
 	
VB.NET method call:
`Public Function GetProcessSchema(ByVal process As String, ByVal version As Integer) As System.Xml.XmlNode`	

Input	
`process`	**Input Type:** System.String **Description:** `process` represents the name of the business process in which the XML schema is required.
`version`	**Input Type:** System.Int32 **Description:** `version` represents the version of the specified business process.
Return Conditions	
`GetProcessSchema` returns the process XML schema in an XMLNode format.	

GetProcessStepQuerySchema

Method: GetProcessStepQuerySchema	

Method description: This method retrieves the resulting XML schema of an XQuery that accesses process and step XML schemas.
 	
C# method call:
`public System.Xml.XmlNode GetProcessStepQuerySchema(string process, int version, string step, string XQuery)`
 	
VB.NET method call:
`Public Function GetProcessStepQuerySchema(ByVal process As String, ByVal version As Integer, ByVal step As String, ByVal XQuery As String) As System.Xml.XmlNode`	

Input	
`process`	**Input Type:** System.String **Description:** `process` represents the name of the business process on which the XQuery is to run.
`version`	**Input Type:** System.Int32 **Description:** `version` represents the version of the specified business process.
`XQuery`	**Input Type:** System.String **Description:** `XQuery` represents an XQuery that accesses both process and step XML schemas.
Return Conditions	
`GetProcessStepQuerySchema` returns the XML schema of the specified XQuery.	

GetProcessStepSchema

Method: GetProcessStepSchema	

Method description: This method retrieves the XML schema that includes a specified process and step XML schema.
 	
C# method call:
`public System.Xml.XmlNode GetProcessStepSchema(string process, int version, string step)`
 	
VB.NET method call:
`Public Function GetProcessStepSchema(ByVal process As String, ByVal version As Integer, ByVal step As String) As System.Xml.XmlNode`	

Input	
`process`	**Input Type:** System.String **Description:** `process` represents the name of the business process in which the XML schema is required.
`version`	**Input Type:** System.Int32 **Description:** `version` represents the version of the specified business process.
`step`	**Input Type:** System.String **Description:** `step` represents the step for which the XML schema is to be returned.
Return Conditions	
`GetProcessStepSchema` returns the XML schema in an XMLNode format.	

RunBPMQuery

Method: RunBPMQuery	

Method description: This method retrieves the data and XML schema of a generic XQuery.
C# method call:
`public System.Xml.XmlNode RunBPMQuery(string xquery, bool use_archived_data)`
VB.NET method call:
`Public Function RunBPMQuery(ByVal xquery As String, ByVal use_archived_data As Boolean) As System.Xml.XmlNode`	

Input	
`XQuery`	**Input Type:** System.String **Description:** `XQuery` represents a generic XQuery.
`use_archived_data`	**Input Type:** Boolean **Description:** `use_archived_data` represents whether archived data should be used against the XQuery. If `use_archived_data` is `TRUE`, then archived data is used; if FALSE, then archived data is not used.
Return Conditions	
`RunBPMQuery` returns the results of the specified XQuery in an XMLNode format.	

RunProcessQuery

Method: RunProcessQuery	

Method description: This method executes an XQuery which can access a specified process XML schema.
C# method call:
`public System.Xml.XmlNode RunProcessQuery(string process, int version, string XQuery, bool use_archived_data)`
VB.NET method call:
`Public Function RunProcessQuery(ByVal process As String, ByVal version As Integer, ByVal XQuery As String, ByVal use_archived_data As Boolean) As System.Xml.XmlNode`	

Input	
`process`	**Input Type:** System.String **Description:** `process` represents the name of the business process on which the XQuery is to run.

Method: RunProcessQuery	
`version`	**Input Type:** System.Int32 **Description:** `version` represents the version of the specified business process.
`XQuery`	**Input Type:** System.String **Description:** `XQuery` represents an XQuery which accesses the process XML schema.
`use_archived_data`	**Input Type:** Boolean **Description:** `use_archived_data` represents whether archived data should be used against the XQuery. If `use_archived_data` is `TRUE`, then archived data is used; if FALSE, then archived data is not used.
Return Conditions	
`RunProcessQuery` returns the results of the specified XQuery in an XMLNode format.	

RunProcessStepQuery

Method: RunProcessStepQuery	

Method description: This method executes an XQuery that can access the XML schemas for a specified process and step.
 	
C# method call:
`public System.Xml.XmlNode RunProcessStepQuery(string process, int version, string step, string xquery, bool use_archived_data)`
 	
VB.NET method call:
`Public Function RunProcessStepQuery(ByVal process As String, ByVal version As Integer, ByVal step As String, ByVal xquery As String, ByVal use_archived_data As Boolean) As System.Xml.XmlNode`	

Input	
`process`	**Input Type:** System.String **Description:** `process` represents the name of the business process on which the XQuery is to run.
`version`	**Input Type:** System.Int32 **Description:** `version` represents the version of the specified business process.

Method: RunProcessStepQuery	
`step`	**Input Type:** System.String **Description:** `step` represents the step label for which its XML schema is required.
`XQuery`	**Input Type:** System.String **Description:** `XQuery` represents an XQuery which accesses the process and step XML schemas.
`use_archived_data`	**Input Type:** Boolean **Description:** `use_archived_data` represents whether archived data should be used against the XQuery. If `use_archived_data` is `TRUE`, then archived data is used; if FALSE, then archived data is not used.
Return Conditions	
`RunProcessStepQuery` returns the results of the specified XQuery in an XMLNode format.	

4

Ultimus EIKService Web service

The Ultimus EIKService Web service is used to write and read to/from existing Ultimus OC (organization chart) structures.

OC Read methods

Outlined below are descriptions of all OC read methods within the Ultimus EIKService Web service (in alphabetical order):

- *GetEmailAddress*
- *GetGroupMembers*
- *GetGroups*
- *GetPrimaryJobFunction*
- *GetUserJobFunctions*

GetEmailAddress

Method: GetEmailAddress	

Method description: This method retrieves a specified user's e-mail address from a business organization. If the user is not found the call fails.
 	
C# method call:
`public string GetEmailAddress(out string strError, string strUserName)`
 	
VB.NET method call:
`Public Function GetEmailAddress(ByRef strError As String, ByVal strUserName As String) As String`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the short name of the user whose e-mail address is to be retrieved.

Method: GetEmailAddress	
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to get the user's e-mail address.
Return Conditions	
`GetEmailAddress` returns the specified user's e-mail address.	

GetGroupMembers

Method: GetGroupMembers	

Method description: This method retrieves the list of members (users only) from a specified group in a business organization. This method ignores any hierarchy within the group.
 	
C# method call:
`public string[] GetGroupMembers(out string strError, string strGroup)`
 	
VB.NET method call:
`Public Function GetGroupMembers(ByRef strError As String, ByVal strGroup As String) As String()`	

Input	
`strGroup`	**Input Type:** System.String **Description:** `strGroup` represents the name of the group whose members are to be retrieved.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to get the group members.
Return Conditions	
`GetGroupMembers` returns a list of group members from the specified group.	

GetGroups

Method: GetGroups	

Method description: This method retrieves all the groups present in the business organization.
C# method call:
`public string[] GetGroups(out string strError)`
VB.NET method call:
`Public Function GetGroups() As String(ByRef strError As String)`	

Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to get the groups.
Return Conditions	
`GetGroups` returns the list of all groups in the business organization.	

GetPrimaryJobFunction

Method: GetPrimaryJobFunction	

Method description: This method retrieves a specified user's primary job function from the business organization.
C# method call:
`public string GetPrimaryJobFunction(out string strError, string strUser)`
VB.NET method call:
`Public Function GetPrimaryJobFunction(ByRef strError As String, ByVal strUser As String) As String`	

Input	
`strUser`	**Input Type:** System.String **Description:** `strUser` represents the short name of the user whose primary job function is to be retrieved.

Method: GetPrimaryJobFunction	
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to get the user's primary job function.
Return Conditions	
`GetPrimaryJobFunction` returns the specified user's primary job function.	

GetUserJobFunctions

Method: GetUserJobFunctions	

Method description: This method retrieves a list of job function(s) assigned to a specified user in the business organization.
 	
C# method call:
`public string[] GetUserJobFunctions(out string strError, string strUser)`
 	
VB.NET method call:
`Public Function GetUserJobFunctions(ByRef strError As String, ByVal strUser As String) As String()`	

Input	
`strUser`	**Input Type:** System.String **Description:** `strUser` represents the short name of the user whose job function(s) are to be retrieved.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to get the user's job functions.
Return Conditions	
`GetUserJobFunctions` returns a list of the specified user's job function(s).	

OC Write methods

Outlined below are descriptions of all OC write methods within the Ultimus EIKService Web service (in alphabetical order):

- *AddGroupMember*
- *DeleteGroup*
- *RemoveGroupMember*
- *SetEmailAddress*
- *SetPrimaryJobFunction*
- *UpdateJobFunctionName*
- *UpdateUserForJobFunction*

AddGroupMember

Method: AddGroupMember	

Method description: This method is used to add a user to a group within a business organization. If the specified group does not exist then a new group with the given name is created.
 	
C# method call:
`public bool AddGroupMember(out string strError, string strGroupName, string strUserName, string strUserFullName)`
 	
VB.NET method call:
`Public Function AddGroupMember(ByRef strError As String, strGroupName As String, strUserName as String, strUserFullName As String) As Boolean`	

Input	
`strGroupName`	**Input Type:** System.String **Description:** `strGroupName` represents the name of the group in which the user is to be added.
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the short name of the user which is to be added in the specified group.
`strUserFullName`	**Input Type:** System.String **Description:** `strUserFullName` represents the full name of the user which is to be added in the specified group.

Method: AddGroupMember	
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to add a group member.
Return Conditions	
`AddGroupMember` returns `TRUE` if the user is added to the specified group; othewise, `AddGroupMember` returns `FALSE`.	

DeleteGroup

Method: DeleteGroup	

Method description: This method deletes the group with its members from a business organization.
C# method call:
`public bool DeleteGroup(out string strError, string strGroupName)`
VB.NET method call:
`Public Function DeleteGroup(ByRef strError as String, ByVal strGroupName As String) As Boolean`	

Input	
`strGroupName`	**Input Type:** System.String **Description:** `strGroupName` represents the name of the group to be deleted.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to delete a group.
Return Conditions	
`DeleteGroup` returns `TRUE` if the specified group is deleted; otherwise `DeleteGroup` returns `FALSE`.	

RemoveGroupMember

Method: RemoveGroupMember	

Method description: This method is used to remove a user from a group in a business organization. If the specified group becomes empty with the execution of this command then the group is also deleted.
 	
C# method call:
`public bool RemoveGroupMember(out string strError, string strGroupName, string strUserName)`
 	
VB.NET method call:
`Public Function RemoveGroupMember(ByRef strError as String, ByVal strGroupName As String, ByVal strUserName As String) As Boolean`	

Input	
`strGroupName`	**Input Type:** System.String **Description:** `strGroupName` represents the name of the group from which the user is to be deleted.
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the short name of the user to be deleted from the specified group.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to remove a group member.
Return Conditions	
`RemoveGroupMember` returns `TRUE` if the specified user is removed from the specified group; otherwise `RemoveGroupMember` returns `FALSE`.	

SetEmailAddress

Method: SetEmailAddress	

Method description: This method sets the e-mail address of a user in the business organization. If the user is not found the call fails.
 	
C# method call:
`public bool SetEmailAddress(out string strError, string strUserName, string strEmailAddress)`
 	
VB.NET method call:
`Public Function SetEmailAddress(ByRef strError as String, ByVal strUserName As String, ByVal strEmailAddress As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the short name of the user whose e-mail address is to be set.
`strEmailAddress`	**Input Type:** System.String **Description:** `strEmailAddress` represents the e-mail address of the specified user.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to set the user's e-mail address.
Return Conditions	
`SetEmailAddress` returns `TRUE` if the specified user's e-mail address is changed successfully; otherwise `SetEmailAddress` returns `FALSE`.	

SetPrimaryJobFunction

Method: SetPrimaryJobFunction	

Method description: This method changes the primary job function of the specified user. The method fails if the specified job function does not exist for the specified user in the specified department.
 	
C# method call:
`public bool SetPrimaryJobFunction(out string strError, string strDepartment, string strUser, string strJobFunction)`
 	
VB.NET method call:
`Public Function SetPrimaryJobFunction(ByRef strError as String, ByVal strDepartment As String, ByVal strUser As String, ByVal strJobFunction As String) As Boolean`	

Input	
`strDepartment`	**Input Type:** System.String **Description:** `strDepartment` represents the name of the department in which to set the user's primary job function.
`strUser`	**Input Type:** System.String **Description:** strUser represents the short name of the user whose primary job function is to be set.
`strJobFunction`	**Input Type:** System.String **Description:** `strJobFunction` represents the job function title which is to be set as the primary job function.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to set the user's primary job function.
Return Conditions	
`SetPrimaryJobFunction` returns `TRUE` if the job function title is properly changed for the specified user; otherwise `SetPrimaryJobFunction` returns `FALSE`.	

UpdateJobFunctionName

Method: UpdateJobFunctionName	

Method description: This method updates the job function title in any department of a business organization.
 	
C# method call:
`public bool UpdateJobFunctionName(out string strError, string strDepartment, string strOldJFName, string strNewJFName)`
 	
VB.NET method call:
`Public Function UpdateJobFunctionName(ByRef strError as String, ByVal strDepartment As String, ByVal strOldJFName As String, ByVal strNewJFName As String) As Boolean`	

Input	
`strDepartment`	**Input Type:** System.String **Description:** `strDepartment` represents the name of the department in which the job function title is to be updated.
`strOldJFName`	**Input Type:** System.String **Description:** `strOldJFName` represents the existing job function title which is to be updated.
`strNewJFName`	**Input Type:** System.String **Description:** `strNewJFName` represents the new job function title.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to update the job function title.
Return Conditions	
`UpdateJobFunctionName` returns `TRUE` if the job function title is properly updated; otherwise `UpdateJobFunctionName` returns `FALSE`.	

UpdateUserForJobFunction

Method: UpdateUserForJobFunction	

Method description: This function changes the user against a job function when the specified job function is the only job function of the specified user. This method throws an exception if the job function is the primary job function of the existing user and there already is another job function assigned to this person.
 	
C# method call:
`public bool UpdateUserForJobFunction(out string strError, string strDepartment, string strJobFunction, string strUser)`
 	
VB.NET method call:
`Public Function UpdateUserForJobFunction(ByRef strError as String, ByVal strDepartment As String, ByVal strJobFunction As String, ByVal strUser As String) As Boolean`	

Input	
`strDepartment`	**Input Type:** System.String **Description:** `strDepartment` represents the name of the department in which the user's job function is to be set.
`strJobFunction`	**Input Type:** System.String **Description:** `strJobFunction` represents the job function title for which user which is to be updated.
`strUser`	**Input Type:** System.String **Description:** `strUser` represents the short name of the user by whom the job function is to be updated.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains any error message returned by the Ultimus engine when it attempts to update the user's job function.
Return Conditions	
`UpdateUserForJobFunction` returns `TRUE` if the specified job function user is properly updated; otherwise `UpdateUserForJobFunction` returns `FALSE`.	

5

Ultimus ClientServices Web service

The Ultimus ClientServices API is new with Ultimus Adaptive BPM Suite 8. This interface is not only utilized by Ultimus Client and Ultimus Thin Client, but also should be utilized when building custom clients. Outlined below are descriptions of all methods within the `ClientServices` namespace (in alphabetical order):

- *AbortIncident*
- *AddGroupView*
- *AddUserView*
- *AddViewFolder*
- *AreAccessRightsEnabled*
- *AssignAllCurrentTasks*
- *AssignQueueTask*
- *AssignTask*
- *AssignTasks*
- *AssignTaskToProcExpert*
- *CanAbort*
- *CanAssignTo*
- *CheckInTask*
- *CheckOutTask*
- *CheckProcessRight*
- *CheckUserRight*
- *DeleteGroupView*
- *DeleteTask*
- *DeleteUserView*
- *DeleteViewFolder*
- *ForwardTasks*

- *GetAdminStatusMessage*
- *GetAllJobFunctionsOfUser*
- *GetAssociates*
- *GetDataByViewID*
- *GetDataForView*
- *GetDepartmentsList*
- *GetDetailedIncidentStatus*
- *GetDomainsOnServer*
- *GetFormURLForEdit*
- *GetFormURLForPreview*
- *GetForwardableTasks*
- *GetForwardableTasksEx*
- *GetForwardedTasks*
- *GetFullNameFromShortName*
- *GetFullnamesFromShortnames*
- *GetGroupView*
- *GetGroupViews*
- *GetGroupsViewsList*
- *GetIncidentData*
- *GetIncidentGraphicalStatus*
- *GetIncidentMemo*
- *GetInitiator*
- *GetInstalledProcesses*
- *GetInstalledProcessesbyLatestVersion*
- *GetNextQueueTask*
- *GetOrgNameFromDomain*
- *GetPersonOrganizations*
- *GetProcessHelpURL*
- *GetProcessSchema*
- *GetProcessSteps*

- *GetReports*
- *GetSSOURL*
- *GetServerVersion*
- *GetStepDefaultForm*
- *GetStepSchema*
- *GetTaskCheckOutUser*
- *GetTaskCheckOutUser*
- *GetTaskData*
- *GetTaskHelpURL*
- *GetTimezoneInformation*
- *GetUserEmailAddress*
- *GetUserFolderList*
- *GetUserGroups*
- *GetUserIDForJobFunction*
- *GetUserNamesforSharedView*
- *GetUserPreferences*
- *GetUserProperties*
- *GetUserSubordinates*
- *GetUserViewsForClient*
- *GetUserViewsList*
- *GetViewByID*
- *GetViewsSharedToUser*
- *IsDepartmentMember*
- *IsUserHierarchicallyAbove*
- *IsValidSessionID*
- *LoginUser*
- *LogoutUser*
- *PutTaskInQueue*
- *RemoveAssociate*
- *RestoreGenericUserViews*

- *ReturnTask*
- *SaveTemplate*
- *SendTask*
- *SetForwardedTasks*
- *SetAssociates*
- *SetUserEmailAddress*
- *SetUserPreferences*
- *ShareViewWithUser*
- *TakeBackTask*
- *UnShareViewWithUser*
- *UpdateGroupView*
- *UpdateUserView*
- *UpdateViewFolder*
- *UpdateViewFolderName*
- *UpdateViewSequenceNumberInFolder*
- *ValidateUser*

AbortIncident

Method: AbortIncident	

Method description: This method aborts the actively loaded incident.
 	
C# method call:
`public bool AbortIncident(string strSessionID, string strProcessName, uint uiIncidentNo, string strAbortReason, out string strErr)`
 	
VB.NET method call:
`Public Function AbortIncident(ByVal strSessionID As String, ByVal strProcessName As String, ByVal uiIncidentNo As UInteger, ByVal strAbortReason As String, ByRef strErr As String) As Boolean`	

Input	
`SessionID`	**Input Type:** System.String **Description:** `SessionID` represents the logged on user's session ID; the user should have the rights to abort the incident.
`strProcessName`	**Input Type:** System.String **Description:** `strProcessName` represents the name of the process whose incident is to be aborted.
`uiIncidentNo`	**Input Type:** Unsigned Integer **Description:** `lIncidentNo` represents the incident number that is to be aborted.
`strAbortReason`	**Input Type:** System.String **Description:** `strAbortReason` represents the reason for aborting the incident.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to abort the incident.
Return Conditions	
`AbortIncident` returns `TRUE` if the incident could be aborted. `AbortIncident` returns `FALSE` if the incident could not be aborted.	

AddGroupView

Method: AddGroupView	

Method description: This method adds the given view to the list of views for the given group.
 	
C# method call:
`public bool AddGroupView(string strSessionID, string strOrgName, string strGroupName, string strViewName, string strViewID, object objViewInfo, out string strErr)`
 	
VB.NET method call:
`Public Function AddGroupView(ByVal strSessionID As String, ByVal strGroupName As String, ByVal strOrgName As String, ByVal strViewName As String, ByVal strViewID As String, ByVal objViewInfo As Object, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to add the view to the specified group.
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` represents the organization name to which the group belongs.
`strGroupName`	**Input Type:** System.String **Description:** `strGroupName` represents the name of the group to be added.
`strViewName`	**Input Type:** System.String **Description:** `strViewName` represents the name of the view.
`strViewID`	**Input Type:** System.String **Description:** `strViewID` represent the ID of the view.
`objViewInfo`	**Input Type:** System.Object **Description:** `objViewInfo` a serialized object of `Ultimus.ClientServices.ViewInfo` class.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to add a group view.

Method: AddGroupView
Return Conditions
`AddGroupView` returns `TRUE` if the specified view can be added to the specified user's group views.
`AddGroupView` returns `FALSE` if the specified view cannot be added to the specified user's group views.

AddUserView

Method: AddUserView	

Method description: This method adds view information to a specified user's list of views.
C# method call:
`public bool AddUserView(string strSessionID, string strViewName, string strViewID, string strFolderID, object, out string strErr)`
VB.NET method call:
`Public Function AddUserView(ByVal strSessionID As String, ByVal strViewName As String, ByVal strViewID As String, ByVal strFolderID As String, ByVal objViewInfo As object, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strViewName`	**Input Type:** System.String **Description:** `strViewName` represents the name of the view to be added.
`strViewID`	**Input Type:** System.String **Description:** `strViewID` represents the ID of the view to be added.
`strFolderID`	**Input Type:** System.String **Description:** `strFolderID` represents the folder ID in which the view is to reside within.
`objViewInfo`	**Input Type:** Object **Description:** `objViewInfo` is a serialized object of `Ultimus.ClientServices.ViewInfo` class.

Method: AddUserView	
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to add the user view.
Return Conditions	
`AddUserView` returns `TRUE` if the specified view may be added to a custom view. `AddUserView` returns `FALSE` if the specified view cannot be added to a custom view.	

AddViewFolder

Method: AddViewFolder	

Method description: This method creates a new folder using a specified name and ID.
C# method call:
`public bool AddViewFolder(string strSessionID, string strFolderName, string strFolderID, out string strErr)`
VB.NET method call:
`Public Function AddViewFolder(ByVal strSessionID As String, ByVal strFolderName As String, ByVal strFolderID As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strFolderName`	**Input Type:** System.String **Description:** `strFolderName` represents the name of the folder that is to be added.
`strFolderID`	**Input Type:** System.String **Description:** `strFolderID` represents the ID that is to be associated with the specified folder name.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to add a folder view.

Method: AddViewFolder
Return Conditions
`AddViewFolder` returns `TRUE` if the specified folder may be added. `AddViewFolder` returns `FALSE` if the specified folder cannot be added.

AreAccessRightsEnabled

Method: AreAccessRightsEnabled	

Method description: This method checks whether the access rights are enabled or not.
 	
C# method call:
`public bool AreAccessRigthsEnabled(out bool bEnabled, out string strErr)`
 	
VB.NET method call:
`Public Function AreAccessRigthsEnabled(ByRef bEnabled As Boolean, ByRef strErr As String) As Boolean`	

Output	
`bEnabled`	**Input Type:** System.Boolean **Description:** `bEnabled` returns `TRUE` when the access rights have been enabled; otherwise, `bEnabled` returns `FALSE`.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to check the access rights.
Return Conditions	
`AreAccessRightsEnabled` returns `TRUE` if the access rights are enabled.	

AssignAllCurrentTasks

Method: AssignAllCurrentTasks	

Method description: This method assigns the logged on client user's current tasks to a specified user.
 	
C# method call:
`public bool AssignAllCurrentTasks(string strSessionID, string strOrgName, string strAssignToUser, out string strErr)`
 	
VB.NET method call:
`Public Function AssignAllCurrentTasks(ByVal strSessionID As String, ByVal strAssignToUser As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` specifies the name of the organization to which the user belongs.
`strAssignToUser`	**Input Type:** System.String **Description:** `strAssignToUser` represents the short name of the user to whom the tasks are to be assigned.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to assign all current tasks.
Return Conditions	
`AssignAllCurrentTasks` returns `TRUE` if the logged on user's tasks are assigned. `AssignAllCurrentTasks` returns `FALSE` if the logged on users's tasks are not assigned.	

AssignQueueTask

Method: AssignQueueTask	

Method description: This method assigns a queue task to a specified user.
 	
C# method call:
`public bool AssignQueueTask(string strSessionID, string strTaskID, string strAssignToUser, out string strErr)`
 	
VB.NET method call:
`Public Function AssignQueueTask(ByVal strSessionID As String, ByVal strTaskID As String, ByVal strAssignToUser As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strTaskID`	**Input Type:** System.String **Description:** `strAssignToUser` specifies the ID of the queue task to be assigned.
`strAssignToUser`	**Input Type:** System.String **Description:** `strAssignToUser` specifies the short name of the user to whom the task is to be assigned.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to assign a queue task.
Return Conditions	
`AssignQueueTask` returns `TRUE` if the queue task is assigned. `AssignQueueTask` returns `FALSE` if the queue task is not assigned.	

AssignTask

Method: AssignTask	

Method description: This method assigns a task to a specified user.
 	
C# method call:
`public bool AssignTask(string strSessionID, string strOrgName, string strTaskID, string strAssignToUser, out string strErr)`
 	
VB.NET method call:
`Public Function AssignTask(ByVal strSessionID As String, ByVal strOrgName As String, ByVal strTaskID As String, ByVal strAssignToUser As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` specifies the organization name.
`strTaskID`	**Input Type:** System.String **Description:** `strTaskID` represents the ID of the task to be assigned.
`strAssignToUser`	**Input Type:** System.String **Description:** `strAssignToUser` represents the short name of the user to whom the task is to be assigned.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to assign a task.
Return Conditions	
`AssignTask` returns `TRUE` if the task is assigned to the specified user. `AssignTask` returns `FALSE` if the task is not assigned to the specified user.	

AssignTasks

Method: AssignTasks	

Method description: This method assigns specific tasks to a specified user.
 	
C# method call:
`public bool AssignTasks(string strSessionID, string strOrgName, TaskToAssign[] listTasksToAssign, out string strErr)`
 	
VB.NET method call:
`Public Function AssignTasks(ByVal strSessionID As String, ByVal strOrgName As String, ByVal listTasksToAssign() As TaskToAssign, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` specifies the organization name.
`listTasksToAssign`	**Input Type:** Object array of class `TaskToAssign` **Description:** `listTasksToAssign` specifies an array of `TaskToAssign` class.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to assign the tasks.
Return Conditions	
`AssignTasks` returns `TRUE` if the tasks are assigned to the specified user. `AssignTasks` returns `FALSE` if the tasks are not assigned to the specified user.	

AssignTaskToProcExpert

Method: AssignTaskToProcExpert	

Method description: This method assigns a task to the Process Expert.
 	
C# method call:
`public bool AssignToProcExpert(string strSessionID, string strTaskID, string strNotes, out string strErr)`
 	
VB.NET method call:
`Public Function AssignToProcExpert(ByVal strSessionID As String, ByVal strTaskID As String, ByVal strNotes As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strTaskID`	**Input Type:** System.String **Description:** `strTaskID` specifies the ID of the task to be assigned.
`strNotes`	**Input Type:** System.String **Description:** `strNotes` specifies the notes or comments associated with the task.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to assign a task to the Process Expert.
Return Conditions	
`AssignTaskToProcExpert` returns `TRUE` if the task is assigned to the Process Expert. `AssignTaskToProcExpert` returns `FALSE` if the task is not assigned to the Process Expert.	

CanAbort

Method: CanAbort	

Method description: This method checks whether the incident of the specified process can be aborted or not.
 	
C# method call:
`public bool CanAbort(string strProcessName, uint lIncidentNo, out bool bAbort, out string strErr)`
 	
VB.NET method call:
`Public Function CanAbort(ByVal strProcessName As String, ByVal lIncidentNo As UInteger, ByRef bAbort As Boolean, ByRef strErr As String) As Boolean`	

Input	
`strProcessName`	**Input Type:** System.String **Description:** `strProcessName` specifies the name of the process whose incident needs to be aborted.
`IIncidentNo`	**Input Type:** System.UInt32 **Description:** `IIncidentNo` specifies the incident number that is to be aborted.
Output	
`bAbort`	**Output Type:** System.Boolean **Description:** `bAbort` returns TRUE if the incident can be aborted; otherwise, `bAbort` returns FALSE.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to check whether the incident can be aborted or not.
Return Conditions	
`CanAbort` returns `TRUE` if the Ultimus engine can check whether the incident may be aborted. `CanAbort` returns `FALSE` if the Ultimus engine cannot check whether the incident may be aborted.	

CanAssignTo

Method: CanAssignTo	

Method description: This method checks if the specified step of a process can be assigned to a specified user or not.
C# method call:
`public bool CanAssignTo(string strUser, string strProcess, string strStepLabel, out string strErr)`
VB.NET method call:
`Public Function CanAssignTo(ByVal strUser As String, ByVal strProcess As String, ByVal strStepLabel As String, ByRef strErr As String) As Boolean`	

Input	
`strUser`	**Input Type:** System.String **Description:** `strUser` specifies the short name of the user to whom the task is to be assigned.
`strProcess`	**Input Type:** System.String **Description:** `strProcess` specifies the name of the process whose task is to be assigned.
`strStepLabel`	**Input Type:** System.String **Description:** `strStepLabel` specifies the step label of the task to be assigned.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to check whether the specified task can be assigned or not.
Return Conditions	
`CanAssignTo` returns `TRUE` if the step may be assigned to the specified user. `CanAssignTo` returns `FALSE` if the step cannot be assigned to the specified user.	

CheckInTask

Method: CheckInTask	

Method description: This method checks-in a checked out task and sets the given XML object in the task.
 	
C# method call:
`public bool CheckInTask(string strSessionID, string strTaskID, object objXML, out string strErr)`
 	
VB.NET method call:
`Public Function CheckInTask(ByVal strSessionID As String, ByVal strTaskID As String, ByVal objXML As Object, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strTaskID`	**Input Type:** System.String **Description:** `strTaskID` specifies the ID of the task to be checked in.
`objXML`	**Input Type:** System.Object **Description:** `objXML` specifies the task data, in XML format, serialized in an object.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to check in a task.
Return Conditions	
`CheckInTask` returns `TRUE` if the Ultimus engine can check in the task. `CheckInTask` returns `FALSE` if the Ultimus engine cannot check in the task.	

CheckOutTask

Method: CheckOutTask	

Method description: This method checks out a checked in task to the logged on user.
 	
C# method call:
`public bool CheckOutTask(string strSessionID, string strTaskID, out string strErr)`
 	
VB.NET method call:
`Public Function CheckOutTask(ByVal strSessionID As String, ByVal strTaskID As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strTaskID`	**Input Type:** System.String **Description:** `strTaskID` specifies the ID of the task to be checked out.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to check out a task.
Return Conditions	
`CheckOutTask` returns `TRUE` if the Ultimus engine can check out the task. `CheckOutTask` returns `FALSE` if the Ultimus engine cannot check out the task.	

CheckProcessRight

Method: CheckProcessRight	

Method description: This method checks a specified user's rights for a specified process.
 	
C# method call:
`public bool CheckProcessRight(ProcessRIGHTS right, string strProcess, string strUserName, out bool bHasRight, out string strErr)`
 	
VB.NET method call:
`Public Function CheckProcessRight(ByVal right As ProcessRIGHTS, ByVal strProcess As String, ByVal strUserName As String, ByRef bHasRight As Boolean, ByRef strErr As String) As Boolean`	

Input	
`Right`	**Input Type:** `ProcessRIGHTS` **Description:** `Right` specifies possible process rights listed under `Ultimus.ClientServices.ProcessRIGHTS` ENUM class.
`strProcess`	**Input Type:** System.String **Description:** `strProcess` specifies the name of the process whose rights are to be checked.
`strUserName`	**Input Type:** System.String **Description:** `strUserName` specifies the short name of the user against whom the rights are to be checked.
`bHasRight`	**Input Type:** System.Boolean **Description:** `bHasRight` represents whether the user has the specified right against the process or not.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to check the process right.
Return Conditions	
`CheckProcessRight` returns `TRUE` if the Ultimus engine can determine the user's right to the process. `CheckProcessRight` returns `FALSE` if the Ultimus engine cannot determine the user's right to the process.	

CheckUserRight

Method: CheckUserRight	

Method description: This method checks the rights of a specified user.
 	
C# method call:
`public bool CheckUserRight(UserRights right, string strUserName, out bool bHasRight, out string strErr)`
 	
VB.NET method call:
`Public Function CheckUserRight(ByVal right As UserRights, ByVal strUserName As String, ByRef bHasRight As Boolean, ByRef strErr As String) As Boolean`	

Input	
`right`	**Input Type:** `UserRights` **Description:** `right` specifies possible rights of a user listed under `UserRights` class.
`strUserName`	**Input Type:** System.String **Description:** `strUserName` specifies the user short name against whom the rights are to be checked.
`bHasRight`	**Input Type:** System.String **Description:** `bHasRight` represents whether the user has the specified right or not.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to check the user's right.
Return Conditions	
`CheckUserRight` returns `TRUE` if the Ultimus engine can determine the user's right. `CheckUserRight` returns `FALSE` if the Ultimus engine cannot determine the user's right.	

DeleteGroupView

Method: DeleteGroupView	

Method description: This method deletes an identified view from the list of views for the given group name.
 	
C# method call:
`public bool DeleteGroupView(string strSessionID, string strOrgName, string strGroupName, string strViewID, out string strErr)`
 	
VB.NET method call:
`Public Function DeleteGroupView(ByVal strSessionID As String, ByVal strOrgName As String, ByVal strGroupName As String, ByVal strViewID As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` specifies the name of the organization.
`strGroupName`	**Input Type:** System.String **Description:** `strUserName` specifies the name of the group whose view is to be deleted.
`strViewID`	**Input Type:** System.String **Description:** `strViewID` specifies the ID of the group view to be deleted.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to delete a group view.
Return Conditions	
`DeleteGroupView` returns `TRUE` if the Ultimus engine can delete the group view. `DeleteGroupView` returns `FALSE` if the Ultimus engine cannot delete the group view.	

DeleteTask

Method: DeleteTask	

Method description: This method deletes the task from Ultimus BPM Database if the task is completed or aborted. (It is recommended that the incident should be completed or aborted before calling this method.)
 	
C# method call:
`public bool DeleteTask(string strSessionID, string strTaskID, out string strErr)`
 	
VB.NET method call:
`Public Function DeleteTask(ByVal SessionID As String, ByVal strTaskID As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strTaskID`	**Input Type:** System.String **Description:** `strTaskID` specifies the task that is to be deleted.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to delete a task.
Return Conditions	
`DeleteTask` returns `TRUE` if the Ultimus engine can delete the task. `DeleteTask` returns `FALSE` if the Ultimus engine cannot delete the task.	

DeleteUserView

Method: DeleteUserView	

Method description: This method deletes a specified user view.
 	
C# method call:
`public bool DeleteUserView(string strSessionID, string strViewID, out string strErr)`
 	
VB.NET method call:
`Public Function DeleteUserView(ByVal strSessionID As String, ByVal strViewID As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strViewID`	**Input Type:** System.String **Description:** `strViewID` specifies the view to be deleted.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to delete a user view.
Return Conditions	
`DeleteUserView` returns `TRUE` if the Ultimus engine can delete the user view. `DeleteUserView` returns `FALSE` if the Ultimus engine cannot delete the user view.	

DeleteViewFolder

Method: DeleteViewFolder	

Method description: This method deletes the folder identified through a folder ID.
 	
C# method call:
`public bool DeleteViewFolder(string strSessionID, string strFolderID, out string strErr)`
 	
VB.NET method call:
`Public Function DeleteViewFolder(ByVal strSessionID As String, ByVal strFolderID As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strFolderID`	**Input Type:** System.String **Description:** `strFolderID` specifies the folder view to be deleted.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to delete a folder view.
Return Conditions	
`DeleteViewFolder` returns `TRUE` if the Ultimus engine can delete the view folder. `DeleteViewFolder` returns `FALSE` if the Ultimus engine cannot delete the view folder.	

ForwardTasks

Method: ForwardTasks	

Method description: This method assigns the logged on client user's future (incoming) tasks to a specified user until a specific date.
 	
C# method call:
`public bool ForwardTasks(string strSessionID, string strAssignToUser, System.DateTime dtAssignFrom, System.DateTime dtAssignUntil, out string strErr)`
 	
VB.NET method call:
`Public Function ForwardTasks(ByVal strSessionID As String, ByVal strAssignToUser As String, ByVal dtAssignFrom As Date, ByVal dtAssignUntil As Date, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strAssignToUser`	**Input Type:** System.String **Description:** `strAssignToUser` specifies the short name of the user all future tasks are to be assigned.
`dtAssignFrom`	**Input Type:** System.DateTime **Description:** `dtAssignFrom` specifies the starting date from which the task are to be assigned.
`dtAssignUntil`	**Input Type:** System.DateTime **Description:** `dtAssignUntil` specifies to which date all future tasks are to be assigned.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to assign all forwarded tasks.
Return Conditions	
`ForwardTasks` returns `TRUE` if the logged on user's tasks are assigned. `ForwardTasks` returns `FALSE` if the logged on users's tasks are not assigned.	

GetAdminStatusMessage

Method: GetAdminStatusMessage	

Method description: This method acquires the status string from the Ultimus System Administrator.
 	
C# method call:
`public bool GetAdminStatusMessage(out string strMessage, out string strErr)`
 	
VB.NET method call:
`Public Function GetAdminStatusMessage(ByRef strMessage As String, ByRef strErr As String) As Boolean`	

Output	
`strMessage`	**Output Type:** System.String **Description:** `strMessage` returns the status message.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to acquire the Ultimus System Administrator status message.
Return Conditions	
`GetAdminStatusMessage` returns `TRUE` if the Ultimus engine can acquire the Ultimus System Administrator status message. `GetAdminStatusMessage` returns `FALSE` if the Ultimus engine cannot acquire the Ultimus System Administrator status message.	

GetAllJobFunctionsOfUser

Method: GetAllJobFunctionsOfUser

Method description: This method acquires all the job functions of a specified user in a specified organization.
 	
C# method call:
`public bool GetAllJobFunctionsOfUser(string strOrgName, string strUserName, out string[] strArrJobFunctions, out string strErr)`
 	
VB.NET method call:
`Public Function GetAllJobFunctionsOfUser(ByVal strOrgName As String, ByVal strUserName As String, ByRef strArrJobFunctions() As String, ByRef strErr As String) As Boolean`

Input

Method: GetAllJobFunctionsOfUser	
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` specifies the Organization name to which the user belongs.
`strUserName`	**Input Type:** System.String **Description:** `strUserName` specifies the user short name whose job functions are to be retrieved.
Output	
`strArrJobFunctions`	**Output Type:** System.String **Description:** `strArrJobFunctions` returns a list of the specified user's job function(s).
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to get the user's job function(s).
Return Conditions	
`GetAllJobFunctionsOfUser` returns `TRUE` if the Ultimus engine can retrieve a specified user's job function(s). `GetAllJobFunctionsOfUser` returns `FALSE` if the Ultimus engine cannot retrieve a specified user's job function(s).	
Comments	
Execution: If an existing, valid user's name and organization is given, `TRUE` should be returned along with a list of the user's job function(s). If `strOrgName` is empty, `FALSE` is returned and the output error message is `Organization name is empty`. If `strUserName` is missing, `FALSE` is returned and the output error message is `User name is missing`. If the value of `strOrgName` is present but invalid, the method returns `FALSE` and the error message returns the exception raised by server components. If the value of `strUserName` is present but invalid, `TRUE` is returned and a zero-length list of job functions is returned.	

GetAssociates

Method: GetAssociates	

Method description: This method retrieves a list of associate(s) whose tasks are to be brought in as part of this view in Ultimus Client.
 	
C# method call:
`public bool GetAssociates(string strUserName, out UserAssociate[] listUserAssociates, out string strErr)`
 	
VB.NET method call:
`Public Function GetAssociates(ByVal strUserName As String, ByRef listUserAssociates() As.UserAssociate, ByRef strErr As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` specifies the short name of the user whose associate(s) are to be retrieved.
Output	
`listUserAssociates`	**Output Type:** `UserAssociate` **Description:** `listUserAssociates` returns an array of `UserAssociate` class, listing the specified user's associate(s).
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to get a list of associate(s).
Return Conditions	
`GetAssociates` returns `TRUE` if the Ultimus engine can retrieve a list of associate(s). `GetAssociates` returns `FALSE` if the Ultimus engine cannot retrieve a list of associate(s).	

GetDataByViewID

Method: GetDataByViewID	

Method description: This method retrieves the data for a user's specified view.
 	
C# method call:
`public bool GetDataByViewID(string strSessionID, string strViewID, out System.Xml.XmlNode xmlNode, out string strErr)`
 	
VB.NET method call:
`Public Function GetDataByViewID(ByVal strSessionID As String, ByVal strViewID As String, ByRef xmlNode As System.Xml.XmlNode, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strViewID`	**Input Type:** System.String **Description:** `strViewID` represents the view's ID in which its data is to be retrieved.
Output	
`xmlNode`	**Output Type:** `System.Xml.XmlNode` **Description:** `xmlNode` returns the view data in an XML node.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to get view data.
Return Conditions	
`GetDataByViewID` returns `TRUE` if the Ultimus engine can retrieve the view data. `GetDataByViewID` returns `FALSE` if the Ultimus engine cannot retrieve the view data.	

GetDataForView

Method: GetDataForView	

Method description: This method returns tasks information in XML format based on an object of class `ViewInfo` serialized into type object. The ViewInfo object can be obtained from an existing view or can be populated by custom code.
 	
C# method call:
`public bool GetDataForView(string strSessionID, object ObjViewInfo, out System.Xml.XmlNode xmlNode, out string strErr)`
 	
VB.NET method call:
`Public Function GetDataForView(ByVal strSessionID As String, ByVal ObjViewInfo As Object, ByRef xmlNode As System.Xml.XmlNode, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`objViewInfo`	**Input Type:** System.Object **Description:** `objViewInfo` specifies an object of view info class.
Output	
`xmlNode`	**Output Type:** `System.Xml.XmlNode` **Description:** `xmlNode` returns the view data in an XML node.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the data of the specified view.
Return Conditions	
`GetDataForView` returns `TRUE` if the Ultimus engine can retrieve data based on the specified serialized object of `ViewInfo`. `GetDataForView` returns `FALSE` if the Ultimus engine cannot retrieve data based on the specified serialized object of `ViewInfo`.	

GetDepartmentsList

Method: GetDepartmentsList	

Method description: This method returns a list of departments that are subdepartments of the department passed to it.
 	
C# method call:
`public bool GetDepartmentsList(string strOrgName, string strDeptID, out string[] strArrDepartmentNames, out string[] DepartmentIDs, out string strErr)`
 	
VB.NET method call:
`Public Function GetDepartmentsList(ByVal strOrgName As String, ByVal strDeptID As String, ByRef strArrDepartmentNames() As String, ByRef DepartmentIDs() As String, ByRef strErr As String) As Boolean`	

Input	
`strDeptID`	**Input Type:** System.String **Description:** `strDeptID` specifies the name of the department.
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` specifies the Organization name to which the department belongs.
Output	
`strArrDepartmentNames`	**Output Type:** System.String **Description:** `strArrDepartmentNames` returns a list of department names.
`DepartmentIDs`	**Output Type:** System.String **Description:** `DepartmentIDs` returns a list of department IDs.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the department list.
Return Conditions	
`GetDepartmentsList` returns `TRUE` if the Ultimus engine can retrieve the list of departments. `GetDepartmentsList` returns `FALSE` if the Ultimus engine cannot retrieve the list of departments.	

GetDetailedIncidentStatus

Method: GetDetailedIncidentStatus	

Method description: This method retrieves the detailed incident status of the specified incident of the process.
 	
C# method call:
`public bool GetDetailedIncidentStatus(string strProcessName, uint lIncidentNo, out IncidentStatus objIncidentStatus, out string strErr)`
 	
VB.NET method call:
`Public Function GetDetailedIncidentStatus(ByVal strProcessName As String, ByVal lIncidentNo As UInteger, ByRef objIncidentStatus As IncidentStatus, ByRef strErr As String) As Boolean`	

Input	
`strProcessName`	**Input Type:** System.String **Description:** `strProcessName` specifies the name of the process.
`IIncidentNo`	**Input Type:** System.UInt32 **Description:** `IIncidentNo` specifies the incident number whose status is to be determined.
Output	
`objIncidentStatus`	**Output Type:** `IncidentStatus` **Description:** `objIncidentStatus` returns an object of `IncidentStatus` class containing the status information.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the incident status.
Return Conditions	
`GetDetailedIncidentStatus` returns `TRUE` if the Ultimus engine can retrieve the incident status. `GetDetailedIncidentStatus` returns `FALSE` if the Ultimus engine cannot retrieve the incident status.	

GetDomainsOnServer

Method: GetDomainsOnServer	

Method description: This method retrieves the list of domains available on the computer hosting Ultimus BPM Server.
 	
C# method call:
`public bool GetDomainsOnServer(out string[] strArrDomains, out string strErr)`
 	
VB.NET method call:
`Public Function GetDomainsOnServer(ByRef strArrDomains() As String, ByRef strErr As String) As Boolean`	

Output	
`strArrDomains`	**Output Type:** System.String **Description:** `strArrDomains` returns the list of available domains.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the domains.
Return Conditions	
`GetDomainsOnServer` returns `TRUE` if the Ultimus engine can retrieve the list of domains. `GetDomainsOnServer` returns `FALSE` if the Ultimus engine cannot retrieve the list of domains.	

GetExactCaseUser

Method: GetExactCaseUser	

Method description: This method returns the exact case of the user's short name (as specified in Ultimus Organization Charts), if the value of `strUserName` is a valid Ultimus Organization Charts user. If the `strUserName` value is not in Ultimus Organization Charts, then `struserWithCase` returns the same value as the supplied `strUserName` value.
 	
C# method call:
`public bool GetExactCaseUser(string strOrgName, string strUserName, out string strUserWithCase, out string strErr)`
 	
VB.NET method call:
`Public Function GetExactCaseUser(ByVal strOrgName As String, ByVal strUserName As String, ByRef strUserWithCase As String, ByRef strErr As String) As Boolean`	

Input	
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` specifies the name of the organization to which the user belongs.
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the user's short name (or log on ID). The format for `strUserName` must be `domain_name/user_short_name`.
Output	
`strUserWithCase`	**Output Type:** System.String **Description:** `strUserWithCase` returns the user's short name value (as stored in Ultimus Organization Charts).
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the exact user case.
Return Conditions	
`GetExactCaseUser` returns `TRUE` if the Ultimus engine can retrieve the user's exact case short name. `GetExactCaseUser` returns `FALSE` if the Ultimus engine cannot retrieve the user's exact case short name.	

GetFormURLForEdit

Method: GetFormURLForEdit	

Method description: This method returns the form URL for the task identified by the given task ID. The task that opens at this URL should be already checked out.
 	
C# method call:
`public bool GetFormURLForEdit(string strSessionID, string strTaskID, out string strOpenFormURL, out string strErr)`
 	
VB.NET method call:
`Public Function GetFormURLForEdit(ByVal strSessionID As String, ByVal strTaskID As String, ByRef strOpenFormURL As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strTaskID`	**Input Type:** System.String **Description:** `strTaskID` represents the task ID of a step whose form URL is to be received.
Output	
`strOpenFormURL`	**Output Type:** System.String **Description:** `strOpenFormURL` returns the URL for the specified step.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to get the form URL.
Return Conditions	
`GetFormURLForEdit` returns `TRUE` if the Ultimus engine can retrieve the form URL. `GetFormURLForEdit` returns `FALSE` if the Ultimus engine cannot retrieve the form URL.	

GetFormURLForPreview

Method: GetFormURLForPreview	

Method description: This method extracts the URL of the task for viewing its form.
 	
C# method call:
`public bool GetFormURLForPreview(string strSessionID, string strTaskID, out string strViewFormURL, out string strErr)`
 	
VB.NET method call:
`Public Function GetFormURLForPreview(ByVal strSessionID As String, ByVal strTaskID As String, ByRef strViewFormURL As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strTaskID`	**Input Type:** System.String **Description:** `strUserName` specifies the task ID of the task whose URL is to be retrieved.
Output	
`strViewFormURL`	**Output Type:** System.String **Description:** `strViewFormURL` represents the task's URL which can be opened in a Web browser.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to get the form URL.
Return Conditions	
`GetFormURL` returns `TRUE` if the Ultimus engine can retrieve the form URL for preview. `GetFormURL` returns `FALSE` if the Ultimus engine cannot retrieve the form URL for preview.	

GetForwardableTasks

Method: GetForwardableTasks	

Method description: This method retrieves a list of a user's future tasks that can come to this user's Inbox and which can be forwarded to other users before coming to his or her's Inbox.
 	
C# method call:
`public bool GetForwardableTasks(string strTaskUser, out ForwardedTask[] arrForwardedTasks, out string strErr)`
 	
VB.NET method call:
`Public Function GetForwardableTasks(ByVal strTaskUser As String, ByRef arrForwardedTasks() As ForwardedTask, ByRef strErr As String) As Boolean`	

Input	
`strTaskUser`	**Input Type:** System.String **Description:** `strUserName` represents the user's short name.
Output	
`arrForwardedTasks`	**Output Type:** `ForwardedTask` **Description:** `arrForwardedTasks` returns a list of user's future tasks that can be forwarded to other users.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the user's future tasks.
Return Conditions	
`GetUserForwardedTask` returns `TRUE` if the Ultimus engine can retrieve the user's forwardable tasks. `GetUserForwardedTask` returns `FALSE` if the Ultimus engine cannot retrieve the user's forwardable tasks.	

GetForwardableTasksEx

Method: GetForwardableTasksEx	

Method description: This method is provided as a functional extension to `GetForwardableTasks` and to include the organizational parameter.
 	
C# method call:
`public bool GetForwardableTasksEx(string strOrgName, string strTaskUser, out ForwardedTask[] arrForwardedTasks, out string strErr)`
 	
VB.NET method call:
`Public Function GetForwardableTasksEx(ByVal strOrgName As String, ByVal strTaskUser As String, ByRef arrForwardedTasks() As ForwardedTask, ByRef strErr As String) As Boolean`	

Input	
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` specifies the name of the organization to which the user belongs.
`strTaskUser`	**Input Type:** System.String **Description:** `strTaskUser` represents the user's short name.
Output	
`arrObjProcStep`	**Output Type:** `ForwardedTask` **Description:** `arrObjProcStep` returns an array of `ForwardedTask` class listing all the future tasks of the specified user.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the user's future tasks.
Return Conditions	
`GetForwardableTasksEx` returns `TRUE` if the Ultimus engine can retrieve the user's future tasks. `GetForwardableTasksEx` returns `FALSE` if the Ultimus engine cannot retrieve the user's future tasks.	

GetForwardedTasks

Method: GetForwardedTasks	

Method description: This method acquires a list of a specified user's forwarded tasks.
 	
C# method call:
`public bool GetForwardedTasks(string strTaskUser, out string[] strArrAssignedToUser, out string[] ProcessName, out string[] StepLabel, out string[] StepID, out System.DateTime[] AssignedFrom, out System.DateTime[] AssignedUntil, out string strErr)`
 	
VB.NET method call:
`Public Function GetForwardedTasks(ByVal strTaskUser As String, ByRef strArrAssignedToUser() As String, ByRef ProcessName() As String, ByRef StepLabel() As String, ByRef StepID() As String, ByRef AssignedFrom() As Date, ByRef AssignedUntil() As Date, ByRef strErr As String) As Boolean`	

Input	
`strTaskUser`	**Input Type:** System.String **Description:** `strTaskUser` specifies the short name of the task owner.
Output	
`strArrAssignedToUser`	**Outut Type:** System.String **Description:** `strArrAssignedToUser` specifies the short name of the users to whom the task was assigned.
`ProcessName`	**Output Type:** System.String **Description:** `ProcessName` returns the name(s) of the process(es) which are associated with the forwarded task(s).
`StepLabel`	**Output Type:** System.String **Description:** `StepLabel` returns the label(s) of the assigned task(s).
`StepID`	**Output Type:** System.String **Description:** `StepID` returns the step ID(s) of the assigned task(s).
`AssignedFrom`	**Output Type:** System.DateTime **Description:** `AssignedFrom` returns the start date(s) of the assigned period.
`AssignedUntil`	**Output Type:** System.DateTime **Description:** `AssignedUntil` returns the end date(s) of the assigned period.

Method: GetForwardedTasks	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to get a list of assigned future task(s).
Return Conditions	
`GetForwardedTasks` returns `TRUE` if the Ultimus engine can retrieve forwarded task(s). `GetForwardedTasks` returns `FALSE` if the Ultimus engine cannot retrieve forwarded task(s).	

GetFullNameFromShortName

Method: GetFullNameFromShortName	

Method description: This method returns the full name against a specified short name. If the short name is not found then the given user name is returned.
 	
C# method call:
`public bool GetFullNameFromShortName(string strUserName, out string strUserFullName)`
 	
VB.NET method call:
`Public Function GetFullNameFromShortName(ByVal strUserName As String, ByRef strUserFullName As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the user's short name or log on ID.
Output	
`strUserFullName`	**Output Type:** System.String **Description:** `strUserFullName` returns the specified user's full name.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to get the user's full name.
Return Conditions	
`GetFullNameFromShortName` returns `TRUE` if the Ultimus engine can retrieve the user's full name. `GetFullNameFromShortName` returns `FALSE` if the Ultimus engine cannot retrieve the user's full name.	

GetFullnamesFromShortnames

Method: GetFullnamesFromShortnames	

Method description: This method gets the list of full names against a given list of short names.
 	
C# method call:
`public bool GetFullNamesFromShortNames(string[] strArrDomain, string[] strArrUserNames, out string[] strArrUserFullNames, out string strErr)`
 	
VB.NET method call:
`Public Function GetFullNamesFromShortNames(ByVal strArrDomain() As String, ByVal strArrUserNames() As String, ByRef strArrUserFullNames() As String, ByRef strErr As String) As Boolean`	

Input	
`arrDomain`	**Input Type:** System.String **Description:** `arrDomain` represents the array of user domains.
`strArrUserNames`	**Input Type:** System.String **Description:** `strArrUserNames` represents the array of user short names.
Output	
`strArrUserFullNames`	**Output Type:** System.String **Description:** `strArrUserFullNames` returns the full names of the specified users.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve full names.
Return Conditions	
`GetFullnamesFromShortnames` returns `TRUE` if the Ultimus engine can retrieve the requested full names. `GetFullnamesFromShortnames` returns `FALSE` if the Ultimus engine cannot retrieve the requested full names.	

GetGroupView

Method: GetGroupView	

Method description: This method returns view information defined against a given group.
 	
C# method call:
`public bool GetGroupView(string strOrgName, string strGroupName, string strViewID, out object objGroupView, out string strErr)`
 	
VB.NET method call:
`Public Function GetGroupView(ByVal strOrgName As String, ByVal strGroupName As String, ByVal strViewID As String, ByRef objGroupView As Object, ByRef strErr As String) As Boolean`	

Input	
`strGroupName`	**Input Type:** System.String **Description:** `strGroupName` represents the name of the group.
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` represents the organization name to which the group belongs.
`strViewID`	**Input Type:** System.String **Description:** `strViewID` represents the ID of the desired view.
Output	
`objGroupView`	**Output Type:** System.Object **Description:** `objGroupView` returns the group view data in a serialized object of `ViewInfo` class.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve group view.
Return Conditions	
`GetGroupView` returns `TRUE` if the Ultimus engine can retrieve the group view. `GetGroupView` returns `FALSE` if the Ultimus engine cannot retrieve the group view.	

GetGroupViews

Method: GetGroupViews	

Method description: This method returns a list of views against a specified group.
 	
C# method call:
`public bool GetGroupViews(string strOrgName, string strGroupName, out object[] listGroupViews, out string strErr)`
 	
VB.NET method call:
`Public Function GetGroupViews(ByVal strOrgName As String, ByVal strGroupName As String, ByRef listGroupViews() As Object, ByRef strErr As String) As Boolean`	

Input	
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` represents the organization name to which the group belongs.
`strGroupName`	**Input Type:** System.String **Description:** `strGroupName` represents the name of the group.
Output	
`listGroupViews`	**Output Type:** System.Object **Description:** `listGroupViews` returns the group views information in an array of serialized objects of `ViewInfo` class.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to get group views.
Return Conditions	
`GetGroupViews` returns `TRUE` if the Ultimus engine can retrieve the group views. `GetGroupViews` returns `FALSE` if the Ultimus engine cannot retrieve the group views.	

GetGroupsViewsList

Method: GetGroupsViewsList	

Method description: This method returns the list of views against a specified group name.
 	
C# method call:
`public bool GetGroupViewsList(string strOrgName, string strGroupName, out View[] listViewNames, out string strErr)`
 	
VB.NET method call:
`Public Function GetGroupViewsList(ByVal strOrgName As String, ByVal strGroupName As String, ByRef listViewNames() As View, ByRef strErr As String) As Boolean`	

Input	
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` represents the organization name to which the group belongs.
`strGroupName`	**Input Type:** System.String **Description:** `strGroupName` represents the name of the group.
Output	
`listViewNames`	**Output Type:** `View` **Description:** `listViewNames` returns basic group views information like View Name, View ID, Folder ID in which the view exists and Filter Mask is an array of `View` class.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the group views list.
Return Conditions	
`GetGroupsViewsList` returns `TRUE` if the Ultimus engine can retrieve the group views list. `GetGroupsViewsList` returns `FALSE` if the Ultimus engine cannot retrieve the group views list.	

GetIncidentData

Method: GetIncidentData	

Method description: This method retrieves the incident data against a specified incident number of a given process.
 	
C# method call:
`public bool GetIncidentData(string strSessionID, string strProcessName, uint uiIncidentNo, out string strXML, out string strErr)`
 	
VB.NET method call:
`Public Function GetIncidentData(ByVal strSessionID As String, ByVal strProcessName As String, ByVal uiIncident As UInteger, ByRef strXML As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strProcessName`	**Input Type:** System.String **Description:** `strProcessName` specifies the name of the process.
`uiIncidentNo`	**Input Type:** System.UInt32 **Description:** `uiIncidentNo` specifies the incident number.
Output	
`strXML`	**Output Type:** System.String **Description:** `strXML` returns the incident data.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the incident data.
Return Conditions	
`GetIncidentData` returns `TRUE` if the Ultimus engine can retrieve the incident data. `GetIncidentData` returns `FALSE` if the Ultimus engine cannot retrieve the incident data.	

GetIncidentGraphicalStatus

Method: GetIncidentGraphicalStatus	

Method description: This method retrieves the incident's graphical status.
 	
C# method call:
`public bool GetIncidentGraphicalStatus(string strProcessName, uint uiIncidentNo, out byte[] bytesGif, out string strErr)`
 	
VB.NET method call:
`Public Function GetIncidentGraphicalStatus(ByVal strProcessName As String, ByVal uiIncidentNo As UInteger, ByRef bytesGif() As Byte, ByRef strErr As String) As Boolean`	

Input	
`strProcessName`	**Input Type:** System.String **Description:** `strProcessName` represents the name of the process.
`uiIncidentNo`	**Input Type:** System.UInt32 **Description:** uiIncidentNo specifies the incident number of the process whose graphical status is required.
Output	
`bytesGif`	**Output Type:** System.Byte **Description:** `bytesGif` returns the graphical status in bytes array.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to get the incident's graphical status.
Return Conditions	
`GetIncidentGraphicalStatus` returns `TRUE` if the Ultimus engine can retrieve the incident's graphical status. `GetIncidentGraphicalStatus` returns `FALSE` if the Ultimus engine cannot retrieve the incident's graphical status.	

GetIncidentMemo

Method: GetIncidentMemo	

Method description: This method gets the incident memo of a specified incident.
 	
C# method call:
`public bool GetIncidentMemo(string strProcessName, uint uiIncidentNo, out IncidentMemo objIncidentMemo, out string strErr)`
 	
VB.NET method call:
`Public Function GetIncidentMemo(ByVal strProcessName As String, ByVal uiIncidentNo As UInteger, ByRef objIncidentMemo As IncidentMemo, ByRef strErr As String) As Boolean`	

Input	
`strProcessName`	**Input Type:** System.String **Description:** `strProcessName` represents the name of the process.
`uiIncidentNo`	**Input Type:** System.UInt32 **Description:** uiIncidentNo specifies the incident number whose incident memo is required.
Output	
`objIncidentMemo`	**Output Type:** `IncidentMemo` **Description:** `objIncidentMemo` returns the memo information in an `IncidentMemo` class object.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to get the incident memo.
Return Conditions	
`GetIncidentMemo` returns `TRUE` if the Ultimus engine can retrieve the incident's memo. `GetIncidentMemo` returns `FALSE` if the Ultimus engine cannot retrieve the incident's memo.	

GetInitiator

Method: GetInitiator	

Method description: This method retrieves the initiator of a specified process incident.
 	
C# method call:
`public bool GetInitiator(string strProcessName, uint dwProcessVersion, uint uiIncident, out string strInitiator, out string strErr)`
 	
VB.NET method call:
`Public Function GetInitiator(ByVal strProcessName As String, ByVal dwProcessVersion As UInteger, ByVal uiIncident As UInteger, ByRef strInitiator As String, ByRef strErr As String) As Boolean`	

Input	
`strProcessName`	**Input Type:** System.String **Description:** `strProcessName` represents the name of the process.
`dwProcessVersion`	**Input Type:** System.UInt32 **Description:** `dwProcessVersion` specifies the process version.
`uiIncident`	**Input Type:** System.UInt32 **Description:** `uiIncident` represents the incident number.
Output	
`strInitiator`	**Output Type:** System.String **Description:** `strInitiator` returns the user name of the person who initiated the incident.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the initiator of the incident.
Return Conditions	
`GetInitiator` returns `TRUE` if the Ultimus engine can retrieve the initiator of the incident. `GetInitiator` returns `FALSE` if the Ultimus engine cannot retrieve the initiator of the incident.	

GetInstalledProcesses

Method: GetInstalledProcesses	

Method description: This method returns a list of all published processes on a specified Ultimus BPM Server.
 	
C# method call:
`public bool GetInstalledProcesses(out InstalledProcess [] listInstalledProcesses, out string strErr)`
 	
VB.NET method call:
`Public Function GetInstalledProcesses(ByRef listInstalledProcesses() As InstalledProcess, ByRef strErr As String) As Boolean`	

Output	
`listInstalledProcesses`	**Output Type:** `InstalledProcess` **Description:** `listInstalledProcesses` returns a list of published Ultimus processes in an array of `InstalledProcess` class.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the published processes.
Return Conditions	
`GetInstalledProcesses` returns `TRUE` if the Ultimus engine can retrieve the published processes. `GetInstalledProcesses` returns `FALSE` if the Ultimus engine cannot retrieve the published processes.	

GetInstalledProcessesbyLatestVersion

Method: GetInstalledProcessesbyLatestVersion	

Method description: This method returns a list of all latest versions of all published processes on Ultimus BPM Server.
 	
C# method call:
`public bool GetInstalledProcessesbyLatestVersion(out InstalledProcess [] listInstalledProcesses, out string strErr)`
 	
VB.NET method call:
`Public Function GetInstalledProcessesbyLatestVersion(ByRef listInstalledProcesses() As InstalledProcess, ByRef strErr As String) As Boolean`	

Output	
`listInstalledProcesses`	**Output Type:** `InstalledProcess` **Description:** `listInstalledProcesses`returns a list of published Ultimus processes in an array of `InstalledProcess` class.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the latest versions of published processes.
Return Conditions	
`GetInstalledProcessesbyLatestVersion` returns `TRUE` if the Ultimus engine can retrieve the latest versions of published processes. `GetInstalledProcessesbyLatestVersion` returns `FALSE` if the Ultimus engine cannot retrieve the latest versions of published processes.	

GetNextQueueTask

Method: GetNextQueueTask	

Method description: This method retrieves the next queue task.
 	
C# method call:
`public bool GetNextQueueTask(string strUserName, out string strErr)`
 	
VB.NET method call:
`Public Function GetNextQueueTask(ByVal strUserName As String, ByRef strErr As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` specifies the user's short name.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to get the next queue task.
Return Conditions	
`GetNextQueueTask` returns `TRUE` if the Ultimus engine can retrieve the next queue task. `GetNextQueueTask` returns `FALSE` if the Ultimus engine cannot retrieve the next queue task.	

GetOrgNameFromDomain

Method: GetOrgNameFromDomain	

Method description: This method returns the organization name acquired from a given domain.
 	
C# method call:
`public bool GetOrgNameFromDomain(string strDomain, out string strOrgName, out string strErr)`
 	
VB.NET method call:
`Public Function GetOrgNameFromDomain(ByVal strDomain As String, ByRef strOrgName As String, ByRef strErr As String) As Boolean`	

Input	
`strDomain`	**Input Type:** System.String **Description:** `strDomain` specifies the name of the domain.
Output	
`strOrgName`	**Output Type:** System.String **Description:** `strOrgName` returns the organization name of the specified domain.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the organization name.
Return Conditions	
`GetOrgNameFromDomain` returns `TRUE` if the Ultimus engine can retrieve the organization name. `GetOrgNameFromDomain` returns `FALSE` if the Ultimus engine cannot retrieve the organization name.	

GetPersonOrganizations

Method: GetPersonOrganizations	

Method description: This method retrieves all the organizations to which a given user belongs.
 	
C# method call:
`public bool GetPersonOrganizations(string strUserName, out string[] strArrOrgNames, out string strErr)`
 	
VB.NET method call:
`Public Function GetPersonOrganizations(ByVal strUserName As String, ByRef strArrOrgNames() As String, ByRef strErr As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` specifies the user's short name.
Output	
`strArrOrgNames`	**Output Type:** System.String **Description:** `strArrOrgNames` returns an array of organizations to which the specified user belongs.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the organizations to which a user belongs.
Return Conditions	
`GetPersonOrganizations` returns `TRUE` if the Ultimus engine can retrieve the organizations to which the user belongs. `GetPersonOrganizations` returns `FALSE` if the Ultimus engine cannot retrieve the organizations to which the user belongs.	

GetProcessHelpURL

Method: GetProcessHelpURL	

Method description: This method gets the Help URL for a specified process incident.
 	
C# method call:
`public bool GetProcessHelpURL(string strProcessName, uint uiIncidentNo, out string strHelpURL, out string strErr)`
 	
VB.NET method call:
`Public Function GetProcessHelpURL(ByVal strProcessName As String, ByVal uiIncidentNo As UInteger, ByRef strHelpURL As String, ByRef strErr As String) As Boolean`	

Input	
`strProcessName`	**Input Type:** System.String **Description:** `strProcessName` specifies the name of the process.
`uiIncidentNo`	**Input Type:** System.UInt32 **Description:** `uiIncidentNo` represents the incident number whose process Help URL is to be retrieved.
Output	
`strHelpURL`	**Output Type:** System.String **Description:** `strHelpURL` returns the process Help URL.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the process Help URL.
Return Conditions	
`GetProcessHelpURL` returns `TRUE` if the Ultimus engine can retrieve the process incident's Help URL. `GetProcessHelpURL` returns `FALSE` if the Ultimus engine cannot retrieve the process incident's Help URL.	

GetProcessSchema

Method: GetProcessSchema	

Method description: This method retrieves the list of process schemas of a given process version.
 	
C# method call:
`public bool GetProcessSchema(string strSessionID, string strProcessName, uint uiProcessVersion, out SchemaFile[] listSchemas, out string strProcessDefaultData, out string strErr)`
 	
VB.NET method call:
`Public Function GetProcessSchema(ByVal strSessionID As String, ByVal strProcessName As String, ByVal uiProcessVersion As UInteger, ByRef listSchemas() As SchemaFile, ByRef strProcessDefaultData As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user.
`strProcessName`	**Input Type:** System.String **Description:** `strProcessName` specifies the name of the process.
`uiProcessVersion`	**Input Type:** System.UInt32 **Description:** `uiProcessVersion` represents the process version number.
Output	
`listSchemas`	**Output Type:** `SchemaFile` **Description:** `listSchemas` returns the schema information in an array of `SchemaFile` class.
`strProcessDefaultData`	**Output Type:** System.String **Description:** `strProcessDefaultData` returns the process data in an XML format.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the process schema information.
Return Conditions	
`GetProcessSchema` returns `TRUE` if the Ultimus engine can retrieve the process schema information. `GetProcessSchema` returns `FALSE` if the Ultimus engine cannot retrieve the process schema information.	

GetProcessSteps

Method: GetProcessSteps	

Method description: This method retrieves a list of steps for a specified process.
 	
C# method call:
`public bool GetProcessSteps(string strProcessName, uint lProcessVersion, out string[] listSteps, out string strErr)`
 	
VB.NET method call:
`Public Function GetProcessSteps(ByVal strProcessName As String, ByVal lProcessVersion As UInteger, ByRef listSteps() As String, ByRef strErr As String) As Boolean`	

Input	
`strProcessName`	**Input Type:** System.String **Description:** `strProcessName` specifies the name of the process.
`lProcessVersion`	**Input Type:** System.UInt32 **Description:** `lProcessVersion` specifies the process version.
Output	
`listSteps`	**Output Type:** System.String **Description:** `listSteps` returns an array of specified process steps.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the list of process steps.
Return Conditions	
`GetProcessSteps` returns `TRUE` if the Ultimus engine can retrieve the list of process steps. `GetProcessSteps` returns `FALSE` if the Ultimus engine cannot retrieve the list of process steps.	

GetReports

Method: GetReports	

Method description: This method retrieves a list of reports.
 	
C# method call:
`public bool GetReports(string strSessionID, string strOrgName, out string[] strReportIDs, out string[] strReportsName, out string strErr)`
 	
VB.NET method call:
`Public Function GetReports(ByVal strSessionID As String, ByVal strOrgName As String, ByRef strReportIDs() As String, ByRef strReportsName() As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user; the user should have the rights to perform the task.
`strOrgName`	**Input Type:** System.UInt32 **Description:** `strOrgName` represents the name of the organization in which the logged on user exists.
Output	
`strReportIDs`	**Output Type:** System.String **Description:** `strReportIDs` returns a list of report IDs.
`strReportNames`	**Output Type:** System.String **Description:** `strReportNames` returns a list of report names.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve reports.
Return Conditions	
`GetReports` returns `TRUE` if the Ultimus engine can retrieve the reports. `GetReports` returns `FALSE` if the Ultimus engine cannot retrieve the reports.	

GetSSOURL

Method: GetSSOURL	

Method description: This method checks whether the SSO is enabled; if true, the method returns the Single Sign On's URL.
 	
C# method call:
`public bool GetSSOURL(out bool bEnabled, out string strSSOURL, out string strErr)`
 	
VB.NET method call:
`Public Function GetSSOURL(ByRef bEnabled As Boolean, ByRef strSSOURL As String, ByRef strErr As String) As Boolean`	

Output	
`bEnabled`	**Output Type:** System.Boolean **Description:** `bEnabled` returns `TRUE` if the Single Sign On is enabled; otherwise, it returns `FALSE`.
`strSSOURL`	**Output Type:** System.String **Description:** `strSSOURL` returns the Single Sign On URL.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the Single Sign On URL.
Return Conditions	
`GetSSOURL` returns `TRUE` if the Ultimus engine can retrieve the Single Sign On URL. `GetSSOURL` returns `FALSE` if the Ultimus engine cannot retrieve the Single Sign On URL.	

GetServerVersion

Method: GetServerVersion	

Method description: This method returns the installed Ultimus BPM Server version.
 	
C# method call:
`public bool GetServerVersion(out int dwVersion, out string strErr)`
 	
VB.NET method call:
`Public Function GetServerVersion(ByRef dwVersion As Integer, ByRef strErr As String) As Boolean`	

Output	
`dwVersion`	**Output Type:** System.Int32 **Description:** `dwVersion` returns the Ultimus BPM Server version.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the Ultimus BPM Server version.
Return Conditions	
`GetServerVersion` returns `TRUE` if the Ultimus engine can retrieve the Ultimus BPM Server version. `GetServerVersion` returns `FALSE` if the Ultimus engine cannot retrieve the Ultimus BPM Server version.	

GetStepDefaultForm

Method: GetStepDefaultForm	

Method description: This method returns the type of the selected default form against a specified step.
 	
C# method call:
`public bool GetStepDefaultForm(string strProcessName, uint nVersion, string strStepID, out string strDefaultForm, out string strErr)`
 	
VB.NET method call:
`Public Function GetStepDefaultForm(ByVal strProcessName As String, ByVal nVersion As UInteger, ByVal strStepID As String, ByRef strDefaultForm As String, ByRef strErr As String) As Boolean`	

Input	
`strProcessName`	**Input Type:** System.String **Description:** `strProcessName` specifies the name of the process.
`nVersion`	**Input Type:** System.UInt32 **Description:** `nVersion` represents the process version number.
`strStepID`	**Input Type:** System.String **Description:** `strStepID` represents the ID of the step whose form is to be retrieved.
Output	
`strDefaultForm`	**Output Type:** System.String **Description:** `strDefaultForm` returns the type of the selected default form.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the step's default form.
Return Conditions	
`GetStepDefaultForm` returns `TRUE` if the Ultimus engine can retrieve the step's default form. `GetStepDefaultForm` returns `FALSE` if the Ultimus engine cannot retrieve the step's default form.	

GetStepSchema

Method: GetStepSchema	

Method description: This method retrieves the schema and step default data of a specified step.
 	
C# method call:
`public bool GetStepSchema(string strSessionID, string strProcessName, uint uiProcessVersion, string strStepName, out SchemaFile[] listSchemas, out string strStepDefaultData, out string strErr)`
 	
VB.NET method call:
`Public Function GetStepSchema(ByVal strSessionID As String, ByVal strProcessName As String, ByVal uiProcessVersion As UInteger, ByVal strStepName As String, ByRef listSchemas() As SchemaFile, ByRef strStepDefaultData As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` specifies the session ID of the logged on user.
`strProcessName`	**Input Type:** System.String **Description:** `strProcessName` specifies the name of the process.
`uiProcessVersion`	**Input Type:** System.UInt32 **Description:** `uiProcessVersion` specifies the version of the process.
`strStepName`	**Input Type:** System.String **Description:** `strProcessName` specifies the name of the step whose schema is to be retrieved.

Method: GetStepSchema	
Output	
`listSchemas`	**Output Type:** `SchemaFile` **Description:** `listSchemas` returns schema information in form of `SchemaFile` class array.
`strStepDefaultData`	**Output Type:** System.String **Description:** `strStepDefaultData` returns the step data in an XML format.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the step XML schema.
Return Conditions	
`GetStepSchema` returns `TRUE` if the Ultimus engine can retrieve the step's XML schema. `GetStepSchema` returns `FALSE` if the Ultimus engine cannot retrieve the step's XML schema.	

GetTaskCheckOutUser

Method: GetTaskCheckOutUser	

Method description: This method retrieves the user who has checked out a specified task.
C# method call:
`public bool GetTaskCheckOutUser(string strTaskID, out string strTaskCheckOutUser, out string strErr)`
VB.NET method call:
`Public Function GetTaskCheckOutUser(ByVal strTaskID As String, ByRef strTaskCheckOutUser As String, ByRef strErr As String) As Boolean`	

Input	
`strTaskID`	**Input Type:** System.String **Description:** `strTaskID` represents the ID of the task.

Method: GetTaskCheckOutUser	
Output	
`strChekOutUser`	**Output Type:** System.String **Description:** `strChekOutUser` returns the user who has checked out a specified task.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the user who has checked out the specified task.
Return Conditions	
`GetTaskCheckOutUser` returns `TRUE` if the Ultimus engine can retrieve the user who has checked out a specified task. `GetTaskCheckOutUser` returns `FALSE` if the Ultimus engine cannot retrieve the user who has checked out a specified task.	

GetTaskData

Method: GetTaskData	

Method description: This method retrieves the data of a specified task.
C# method call:
`public bool GetTaskData(string strSessionID, string strTaskID, out string strXML, out string strErr)`
VB.NET method call:
`Public Function GetTaskData(ByVal strSessionID As String, ByVal strTaskID As String, ByVal strXML As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strTaskID`	**Input Type:** System.String **Description:** `strTaskID` represents the ID of the task.

Method: GetTaskData	
Output	
`strXML`	**Output Type:** System.String **Description:** `strXML` returns the task data in XML format.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the task data.
Return Conditions	
`GetTaskData` returns `TRUE` if the Ultimus engine can retrieve the task data. `GetTaskData` returns `FALSE` if the Ultimus engine cannot retrieve the task data.	

GetTaskHelpURL

Method: GetTaskHelpURL	

Method description: This method retrieves the Help URL of a specified task.
C# method call:
`public bool GetTaskHelpURL(string strTaskID, out string strHelpURL, out string strErr)`
VB.NET method call:
`Public Function GetTaskHelpURL(ByVal strTaskID As String, ByRef strHelpURL As String, ByRef strErr As String) As Boolean`	

Input	
`strTaskID`	**Input Type:** System.String **Description:** `strTaskID` represents the ID of the task.
Output	
`strHelpURL`	**Output Type:** System.String **Description:** `strHelpURL` returns the task Help URL of a specified task.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the task Help URL.

Method: GetTaskHelpURL
Return Conditions
`GetTaskHelpURL` returns `TRUE` if the Ultimus engine can retrieve the task Help URL.
`GetTaskHelpURL` returns `FALSE` if the Ultimus engine cannot retrieve the task Help URL.

GetTimezoneInformation

Method: GetTimezoneInformation	

Method description: This method returns time zone information at the computer hosting Ultimus BPM Server in terms of the variance with standard time zone.
 	
C# method call:
`public bool GetTimezoneInformation(out int lBias, out int lStandardBias, out int lDaylightBias, out string strErr)`
 	
VB.NET method call:
`Public Function GetTimezoneInformation(ByRef lBias As Integer, ByRef lStandardBias As Integer, ByRef lDaylightBias As Integer, ByRef strErr As String) As Boolean`	

Output	
`lBias`	**Output Type:** System.Int32 **Description:** `lBias` returns the current bias, in minutes, for local time translation on this computer.
`lStandardBias`	**Output Type:** System.Int32 **Description:** `lStandardBias` returns a bias value that is used during local time translations that occur during standard time.
`lDaylightBias`	**Output Type:** System.Int32 **Description:** `lDaylightBias` returns a bias value that is used during local time translations that occur during daylight time.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the time zone information.
Return Conditions	
`GetTimezoneInformation` returns `TRUE` if the Ultimus engine can retrieve the time zone information.	
`GetTimezoneInformation` returns `FALSE` if the Ultimus engine cannot retrieve the time zone information.	

GetUserEmailAddress

Method: GetUserEmailAddress	

Method description: This method retrieves the e-mail address of a specified user.
 	
C# method call:
`public bool GetUserEmailAddress(string strUserName, out string strUserEmail, out string strErr)`
 	
VB.NET method call:
`Public Function GetUserEmailAddress(ByVal strUserName As String, ByRef strUserEmail As String, ByRef strErr As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the user's short name.
Output	
`strUserEmail`	**Output Type:** System.String **Description:** `strUserEmail` returns the e-mail address of the specified user.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the user's e-mail address.
Return Conditions	
`GetUserEmailAddress` returns `TRUE` if the Ultimus engine can retrieve the user's e-mail address. `GetUserEmailAddress` returns `FALSE` if the Ultimus engine cannot retrieve the user's e-mail address.	

GetUserFolderList

Method: GetUserFolderList	

Method description: This method returns a list of folders defined against a specified user.
 	
C# method call:
`public bool GetUserFolderList(string strUserName, out ViewFolder [] FolderList, out string strErr)`
 	
VB.NET method call:
`Public Function GetUserFolderList(ByVal strUserName As String, ByRef FolderList() As ViewFolder, ByRef strErr As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the user's short name.
Output	
`FolderList`	**Output Type:** `ViewFolder` **Description:** `FolderList` returns an array of `ViewFolder` class containing all the folders of the specified user.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the user's folder list.
Return Conditions	
`GetUserFolderList` returns `TRUE` if the Ultimus engine can retrieve the user's folder list. `GetUserFolderList` returns `FALSE` if the Ultimus engine cannot retrieve the user's folder list.	

GetUserGroups

Method: GetUserGroups	

Method description: This method returns the group(s) to which a specified user belongs.
 	
C# method call:
`public bool GetUserGroups(string strUserName, out GroupInfo[] arrGroups, out string strErr)`
 	
VB.NET method call:
`Public Function GetUserGroups(ByVal strUserName As String, ByRef arrGroups() As GroupInfo, ByRef strErr As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` specifies the user's short name.
Output	
`arrGroups`	**Output Type:** `GroupInfo` **Description:** `arrGroups` returns an array of specified user group(s) as `GroupInfo` class.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the user's group(s).
Return Conditions	
`GetUserGroups` returns `TRUE` if the Ultimus engine can retrieve the user's group(s). `GetUserGroups` returns `FALSE` if the Ultimus engine cannot retrieve the user's group(s).	

GetUserIDForJobFunction

Method: GetUserIDForJobFunction	

Method description: This method retrieves the user ID against a specified primary job function in a given organization.
 	
C# method call:
`public bool GetUserIDForJobFunction(string strOrgName, string strJF, out string strUserID, out string strErr)`
 	
VB.NET method call:
`Public Function GetUserIDForJobFunction(ByVal strOrgName As String, ByVal strJF As String, ByRef strUserID As String, ByRef strErr As String) As Boolean`	

Input	
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` specifies the organization's name to which the user belongs.
`strJF`	**Input Type:** System.String **Description:** `strJF` specifies the user's primary job function.
Output	
`strUserID`	**Output Type:** System.String **Description:** `strUserID` returns the user ID from the specified job function.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the user's ID for the specified primary job function.
Return Conditions	
`GetUserIDForJobFunction` returns `TRUE` if the Ultimus engine can retrieve the user's ID for the specified job function. `GetUserIDForJobFunction` returns `FALSE` if the Ultimus engine cannot retrieve the user's ID for the specified job function.	

GetUserNamesforSharedView

Method: GetUserNamesforSharedView	

Method description: This method returns a list of users with whom a specified view is shared.
 	
C# method call:
`public bool GetUserNamesforSharedView(string strSessionID, string strViewName, string strViewID, out string[] listUsers, out string strErr)`
 	
VB.NET method call:
`Public Function GetUserNamesforSharedView(ByVal strSessionID As String, ByVal strViewName As String, ByVal strViewID As String, ByRef listUsers() As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` specifies the session ID of the logged on user.
`strViewName`	**Input Type:** System.String **Description:** `strViewName` specifies the name of the shared view.
`strViewID`	**Input Type:** System.String **Description:** `strViewID` specifies the ID of the shared view.
Output	
`listUsers`	**Output Type:** System.String **Description:** `listUsers` returns a list of users for the specified shared view.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the user names for the shared view.
Return Conditions	
`GetUserNamesforSharedView` returns `TRUE` if the Ultimus engine can retrieve the user names for the shared view. `GetUserNamesforSharedView` returns `FALSE` if the Ultimus engine cannot retrieve the user names for the shared view.	

GetUserPreferences

Method: GetUserPreferences	

Method description: This method retrieves a specified user's preferences.
 	
C# method call:
`public bool GetUserPreferences(string strUserName, out object objUserPreferences, out string strErr)`
 	
VB.NET method call:
`Public Function GetUserPreferences(ByVal strUserName As String, ByRef objUserPreferences As Object, ByRef strErr As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` specifies the user's short name.
Output	
`objUserPreferences`	**Output Type:** System.Object **Description:** `objUserPreferences` returns the user preference information in an object of ClientServices.UserPreferences.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the user's preferences.
Return Conditions	
`GetUserPreferences` returns `TRUE` if the Ultimus engine can retrieve the user's preferences. `GetUserPreferences` returns `FALSE` if the Ultimus engine cannot retrieve the user's preferences.	

GetUserProperties

Method: GetUserProperties	

Method description: This method retrieves the specified user properties.
 	
C# method call:
`public bool GetUserProperties(string strOrgName, string strUserName, out UserProperties objUserProperties, out string strErr)`
 	
VB.NET method call:
`Public Function GetUserProperties(ByVal strOrgName As String, ByVal strUserName As String, ByRef objUserProperties As UserProperties, ByRef strErr As String) As Boolean`	

Input	
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` represents the organization's name to which the user belongs.
`strUserName`	**Input Type:** System.String **Description:** `strUserName` specifies the user's short name.
Output	
`objUserProperties`	**Output Type:** `UserProperties` **Description:** `objUserPreferences` returns the user properties in an object of `UserProperties` class.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the user properties.
Return Conditions	
`GetUserProperties` returns `TRUE` if the Ultimus engine can retrieve the user properties. `GetUserProperties` returns `FALSE` if the Ultimus engine cannot retrieve the user properties.	

GetUserSubordinates

Method: GetUserSubordinates	

Method description: This method acquires the subordinates of a specified user in a specified department.
 	
C# method call:
`public bool GetUserSubordinates(string strOrgName, string strUserName, string strDept, out UserSubordinate[] listSubOrdinates, out string strErr)`
 	
VB.NET method call:
`Public Function GetUserSubordinates(ByVal strOrgName As String, ByVal strUserName As String, ByVal strDept As String, ByRef listSubOrdinates() As UserSubordinate, ByRef strErr As String) As Boolean`	

Input	
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` represents the organization's name to which the user belongs.
`strUserName`	**Input Type:** System.String **Description:** `strUserName` specifies the user's short name.
`strDept`	**Input Type:** System.String **Description:** `strDept` specifies the name of the department to which the user belongs.
Output	
`listSubOrdinates`	**Output Type:** `UserSubordinate` **Description:** `listSubOrdinates` returns the list of subordinates in a `UserSubordinate` class array.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to acquire the user's subordinates.
Return Conditions	
`GetUserSubordinates` returns `TRUE` if the Ultimus engine can retrieve the user's subordinates. `GetUserSubordinates` returns `FALSE` if the Ultimus engine cannot retrieve the user's subordinates.	

GetUserViewsForClient

Method: GetUserViewsForClient	

Method description: This method returns all the views for which a specified user is the owner (or a part of the list of owners).
 	
C# method call:
`public bool GetUserViewsForClient(string strSessionID, string strOrgName, out object[] listUserViews, out string strErr)`
 	
VB.NET method call:
`Public Function GetUserViewsForClient(ByVal strSessionID As String, ByVal strOrgName As String, ByRef listUserViews() As Object, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` represents the name of the organization.
Output	
`listUserViews`	**Output Type:** System.Object **Description:** `listUserViews` returns the view information in an object array.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve user views for client.
Return Conditions	
`GetUserViewsForClient` returns `TRUE` if the Ultimus engine can retrieve the user views. `GetUserViewsForClient` returns `FALSE` if the Ultimus engine cannot retrieve the user views.	

GetUserViewsList

Method: GetUserViewsList	

Method description: This method returns a list of view objects defined against a specified user.
 	
C# method call:
`public bool GetUserViewsList(string strUserName, out View[] listViewNames, out string strErr)`
 	
VB.NET method call:
`Public Function GetUserViewsList(ByVal strUserName As String, ByRef listViewNames() As View, ByRef strErr As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` specifies the user's short name.
Output	
`listViewNames`	**Output Type:** `View` **Description:** `listViewNames` returns the view list of the specified user in a `View` class array.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the user view list.
Return Conditions	
`GetUserViewsList` returns `TRUE` if the Ultimus engine can retrieve the user view list. `GetUserViewsList` returns `FALSE` if the Ultimus engine cannot retrieve the user view list.	

GetViewByID

Method: GetViewByID	

Method description: This method retrieves the user view information against a specified view ID.
 	
C# method call:
`public bool GetViewByID(string strViewID, out object objViewInfo, out string strErr)`
 	
VB.NET method call:
`Public Function GetViewByID(ByVal strViewID As String, ByRef objViewInfo As Object, ByRef strErr As String) As Boolean`	

Input	
`strViewID`	**Input Type:** System.String **Description:** `strUserName` specifies the ID of the required view.
Output	
`objViewInfo`	**Output Type:** System.Object **Description:** `objViewInfo` returns a serialized object of `ViewInfo` class.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the view by ID.
Return Conditions	
`GetViewByID` returns `TRUE` if the Ultimus engine can retrieve the user view information. `GetViewByID` returns `FALSE` if the Ultimus engine cannot retrieve the user view information.	

GetViewsSharedToUser

Method: GetViewsSharedToUser	

Method description: This method returns a list of objects of all views shared to the logged on user.
 	
C# method call:
`public bool GetViewsSharedToUser(string strSessionID, out object[] listSharedViews, out string strErr)`
 	
VB.NET method call:
`Public Function GetViewsSharedToUser(ByVal strSessionID As String, ByRef listSharedViews() As Object, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the logged on user's session ID; the user should have the rights to perform the task.
Output	
`listSharedViews`	**Output Type:** System.Object **Description:** `listSharedViews` returns the view information in an object array.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to retrieve the shared views.
Return Conditions	
`GetViewsSharedToUser` returns `TRUE` if the Ultimus engine can retrieve the shared views. `GetViewsSharedToUser` returns `FALSE` if the Ultimus engine cannot retrieve the shared views.	

IsDepartmentMember

Method: IsDepartmentMember	

Method description: The method returns a boolean value to indicate whether the specified user is a member of the department or not. The department can be identified by means of its name or identifier.
 	
C# method call:
`public bool IsDepartmentMember(string strOrgName, string strUserName, string strDepartmentName, string strDepartmentID, out bool bResult, out string strErr)`
 	
VB.NET method call:
`Public Function IsDepartmentMember(ByVal strOrgName As String, ByVal strUserName As String, ByVal strDepartmentName As String, ByVal strDepartmentID As String, ByRef bResult As Boolean, ByRef strErr As String) As Boolean`	

Input	
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` specifies the name of the organization.
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the user's short name.
`strDepartmentName`	**Input Type:** System.String **Description:** `strDepartmentName` specifies the name of the department.
`strDepartmentID`	**Input Type:** System.String **Description:** `strDepartmentID` specifies the ID of the specified department.
Output	
`bResult`	**Output Type:** System.Boolean **Description:** `bResult` returns `TRUE` if the user is a member of the specified department; otherwise, it returns `FALSE`.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to check the department member.
Return Conditions	
`IsDepartmentMember` returns `TRUE` if the Ultimus engine can check whether the specified user is a member of the specified department. `IsDepartmentMember` returns `FALSE` if the Ultimus engine cannot check whether the specified user is a member of the specified department.	

IsUserHierarchicallyAbove

Method: IsUserHierarchicallyAbove	

Method description: This method checks whether the given user is hierarchically above a specified user.
 	
C# method call:
`public bool IsUserHierarchicallyAbove(string strOrgName, string strUserName, string strUserHierarchicallyAbove, bool bCheckPrimary, out bool bResult, out string strErr)`
 	
VB.NET method call:
`Public Function IsUserHierarchicallyAbove(ByVal strOrgName As String, ByVal strUserName As String, ByVal strUserHierarchicallyAbove As String, ByVal bCheckPrimary As Boolean, ByRef bResult As Boolean, ByRef strErr As String) As Boolean`	

Input	
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` represents the organization's name to which the user belongs.
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the user's short name.
`strUserHierarchically Above`	**Input Type:** System.String **Description:** `strUserHierarchicallyAbove` specifies the short name of the specified user's supervisor.
`bCheckPrimary`	**Input Type:** System.Boolean **Description:** `bCheckPrimary` is `TRUE` if the primary job function is to be considered for either `strUserName` or `strUserHierarchically Above`.

Method: IsUserHierarchicallyAbove	
Output	
`bResult`	**Output Type:** System.Boolean **Description:** `bResult` returns `TRUE` if the user is the supervisor to `strUserName`; otherwise, it returns `FALSE`.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to check for the user supervisor.
Return Conditions	
`IsUserHierarchicallyAbove` returns `TRUE` if the Ultimus engine can check for the user supervisor. `IsUserHierarchicallyAbove` returns `FALSE` if the Ultimus engine cannot check for the user supervisor.	

IsValidSessionID

Method: IsValidSessionID	

Method description: This method checks if the given session ID is valid or not.
C# method call:
`public bool IsValidSessionID(string strSessionID, string strUser)`
VB.NET method call:
`Public Function IsVaidSessionID(ByVal strSessionID As String, ByVal strUser As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user.
`strUserName`	**Input Type:** System.String **Description:** `strUserName` represents the user's short name.
Return Conditions	
`IsValidSessionID` returns `TRUE` if the session ID is valid. `IsValidSessionID` returns `FALSE` if the session ID is not valid.	

LoginUser

Method: LoginUser	

Method description: This method establishes a client session of the specified user.
 	
C# method call:
`public bool LoginUser(string strDomain, string strUser, string strPassword, out string strSessionID, out string strErr)`
 	
VB.NET method call:
`Public Function LoginUser(ByVal strDomain As String, ByVal strUser As String, ByVal strPassword As String, ByRef strSessionID As String, ByRef strErr As String) As Boolean`	

Input	
`strDomain`	**Input Type:** System.String **Description:** `strDomain` specifies the domain name to which the user belongs.
`strUser`	**Input Type:** System.String **Description:** `strUser` specifies the user's name.
`strPassword`	**Input Type:** System.String **Description:** `strPassword` specifies the user's password.
Output	
`strSessionID`	**Output Type:** System.String **Description:** `strSessionID` returns the session ID of the logged on user.
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to log on the user.
Return Conditions	
`LoginUser` returns `TRUE` if the Ultimus engine can log on the user. `LoginUser` returns `FALSE` if the Ultimus engine cannot log on the user.	

LogoutUser

Method: LogoutUser	

Method description: This method disconnects the logged on session.
 	
C# method call:
`public bool LogoutUser(string strSessionID, out string strErr)`
 	
VB.NET method call:
`Public Function LogoutUser(ByVal strSessionID As String, ByRef strErr As String) As Boolean`	

Input	
`strSession`	**Input Type:** System.String **Description:** `strSession` represents the session ID of the logged on user.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to log off the user.
Return Conditions	
`LogoutUser` returns `TRUE` if the Ultimus engine can log off the user. `LogoutUser` returns `FALSE` if the Ultimus engine cannot log off the user.	

PutTaskInQueue

Method: PutTaskInQueue	

Method description: This method places a task back into queue which has been taken out of the task queue by a specified task ID.
 	
C# method call:
`public bool PutTaskInQueue(string strSessionID, string strTaskID, out string strErr)`
 	
VB.NET method call:
`Public Function PutTaskInQueue(ByVal strSessionID As String, ByVal strTaskId As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user.
`strTaskID`	**Input Type:** System.String **Description:** `strTaskID` represents the ID of the task that is to be put into the queue.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to put the task in queue.
Return Conditions	
`PutTaskInQueue` returns `TRUE` if the Ultimus engine can place the task in queue. `PutTaskInQueue` returns `FALSE` if the Ultimus engine cannot place the task in queue.	

RemoveAssociate

Method: RemoveAssociate	

Method description: This method removes a specified user name from the list of associates to the logged on user.
 	
C# method call:
`public bool RemoveAssociate(string strSessionID, string strAssociate, out string strErr)`
 	
VB.NET method call:
`Public Function RemoveAssociate(ByVal strSessionID As String, ByVal strAssociate As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user; the logged on user should have the necessary rights to perform the current task.
`strAssociate`	**Input Type:** System.String **Description:** `strAssociate` specifies the user short name that is to be removed from the associate list.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to remove an associate.
Return Conditions	
`RemoveAssociate` returns `TRUE` if the Ultimus engine can remove the user from the associate list. `RemoveAssociate` returns `FALSE` if the Ultimus engine cannot remove the user from the associate list.	

RestoreGenericUserViews

Method: RestoreGenericUserViews	

Method description: This method creates the four default views and deletes any and all pre-existing views, both generic and custom.
 	
C# method call:
`public bool RestoreGenericUserViews(string strSessionID, out string strErr)`
 	
VB.NET method call:
`Public Function RestoreGenericUserViews(ByVal strSessionID As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user; the logged on user should have the necessary rights to perform the current task.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to restore the generic user views.
Return Conditions	
`RestoreGenericUserViews` returns `TRUE` if the Ultimus engine can restore the generic user views. `RestoreGenericUserViews` returns `FALSE` if the Ultimus engine cannot restore the generic user views.	

ReturnTask

Method: ReturnTask	

Method description: This method returns a given task with XML data provided.
 	
C# method call:
`public bool ReturnTask(string strSessionID, string strTaskID, string strSummary, string strMemo, string strXML, bool bValidateXML, bool bDisableAbort, uint uiPriority, out string strErr)`
 	
VB.NET method call:
`Public Function ReturnTask(ByVal strSessionID As String, ByVal strTaskID As String, ByVal strSummary As String, ByVal strMemo As String, ByVal strXML As String, ByVal bValidateXML As Boolean, ByVal bDisableAbort As Boolean, ByVal uiPriority As UInteger, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user; the logged on user should have the necessary rights to perform the current task.
`strTaskID`	**Input Type:** System.String **Description:** `strTaskID` represents the ID of the task that is to be returned.
`strSummary`	**Input Type:** System.String **Description:** `strSummary` represents the task summary.
`strMemo`	**Input Type:** System.String **Description:** `strMemo` represents the memo of the specified task.
`strXML`	**Input Type:** System.String **Description:** `strXML` represents the task data in an XML format.
`bValidateXML`	**Input Type:** System.Boolean **Description:** `bValidateXML` validates the given XML if the boolean value is `TRUE`.
`bDisableAbort`	**Input Type:** System.Boolean **Description:** `bDisableAbort` sets an Ultimus keyword named `NoAbort` which is used to determine whether an incident can be aborted or not.
`uiPriority`	**Input Type:** System.UInt32 **Description:** `uiPriority` specifies the priority of the given task.

Method: ReturnTask	
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to return the specified task.
Return Conditions	
`ReturnTask` returns `TRUE` if the Ultimus engine can return the task. `ReturnTask` returns `FALSE` if the Ultimus engine cannot return the task.	

SaveTemplate

Method: SaveTemplate	

Method description: This method saves a template of the given Begin step task with the provided XML data.
C# method call:
`public bool SaveTemplate(string strSessionID, string strTaskID, string strSummary, string strXML, out string strError)`
VB.NET method call:
`Public Function SaveTemplate(ByVal strSessionID As String, ByVal strTaskID As String, ByVal strSummary As String, ByVal strXML As String, ByRef strError As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user; the logged on user should have the necessary rights to perform the current task.
`strTaskID`	**Input Type:** System.String **Description:** `strTaskID` specifies the ID of the task.
`strSummary`	**Input Type:** System.String **Description:** `strSummary` specifies the summary of the task.
`strXML`	**Input Type:** System.String **Description:** `strXML` specifies the data of the task in an XML format.

Method: SaveTemplate	
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to save the task as a template.
Return Conditions	
`SaveTemplate` returns `TRUE` if the Ultimus engine can save the task as a template. `SaveTemplate` returns `FALSE` if the Ultimus engine cannot save the task as a template.	

SendTask

Method: SendTask	

Method description: This method submits a given task with XML data provided.
C# method call:
`public bool SendTask(string strSessionID, string strTaskID, string strSummary, string strMemo, string strXML, bool bValidateXML, bool bDisableAbort, uint uiPriority, out string strErr)`
VB.NET method call:
`Public Function SendTask(ByVal strSessionID As String, ByVal strTaskID As String, ByVal strSummary As String, ByVal strMemo As String, ByVal strXML As String, ByVal bValidateXML As Boolean, ByVal bDisableAbort As Boolean, ByVal uiPriority As UInteger, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` specifies the ID of t he task that needs to be submitted.
`strTaskID`	**Input Type:** System.String **Description:** `strTaskID` specifies the ID of the task.
`strSummary`	**Input Type:** System.String **Description:** `strSummary` specifies the summary of the task.
`strMemo`	**Input Type:** System.String **Description:** `strMemo` specifies the memo of the task.
`strXML`	**Input Type:** System.String **Description:** `strXML` specifies the data of the task in an XML format.
`bValidateXML`	**Input Type:** System.Boolean **Description:** `bValidateXML` validates the given XML if the boolean value is `TRUE`.
`bDisableAbort`	**Input Type:** System.Boolean **Description:** `bDisableAbort` sets an Ultimus keyword named `NoAbort` which is used to determine whether an incident can be aborted or not.
`uiPriority`	**Input Type:** System.UInt32 **Description:** `uiPriority` specifies the priority of the given task.

Method: SendTask	
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to submit the given task.
Return Conditions	
`SendTask` returns `TRUE` if the Ultimus engine can submit the task. `SendTask` returns `FALSE` if the Ultimus engine cannot submit the task.	

SetAssociates

Method: SetAssociates	

Method description: This method enables the calling application to set the users in a list as the logged on user's associates.
 	
C# method call:
`public bool SetAssociates(string strSessionID, UserAssociate[] listUserAssociates, out string strErr)`
 	
VB.NET method call:
`Public Function SetAssociates(ByVal strSessionID As String, ByVal listUserAssociates() As UserAssociate, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user; the logged on user should have the necessary rights to perform the current task.
`listUserAssociates`	**Input Type:** `UserAssociate` **Description:** `listUserAssociates` is an object array of `UserAssociate` class listing the associates.

Method: SetAssociates	
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to set the associates.
Return Conditions	
`SetAssociates` returns `TRUE` if the Ultimus engine can set the users in a list as the logged on user's associates. `SetAssociates` returns `FALSE` if the Ultimus engine cannot set the users in a list as the logged on user's associates.	

SetForwardedTasks

Method: SetForwardedTasks	

Method description: This method assigns the forwarded tasks to a specified user.
 	
C# method call:
`public bool SetForwardedTasks(string strSessionID, string[] AssignedToUser, string[] ProcessName, string[] StepLabel, string[] StepID, System.DateTime[] AssignedFrom, System.DateTime[] AssignedUntil, out string strErr)`
 	
VB.NET method call:
`Public Function SetForwardedTasks(ByVal strSessionID As String, ByVal AssignedToUser() As String, ByVal ProcessName() As String, ByVal StepLabel() As String, ByVal StepID() As String, ByVal AssignedFrom() As Date, ByVal AssignedUntil() As Date, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user; the logged on user should have the necessary rights to perform the current task.
`AssignedToUser`	**Input Type:** System.String **Description:** `AssignedToUser` specifies an array of user short names as associates.
`ProcessName`	**Input Type:** System.String **Description:** `ProcessName` specifies the name of the process(es) in which future tasks are to be assigned.
`StepLabel`	**Input Type:** System.String **Description:** `StepLabel` specifies the list of step label(s) that are to be forwarded.
`StepID`	**Input Type:** System.String **Description:** `StepID` specifies the ID of the specified step(s).
`AssignedFrom`	**Input Type:** System.DateTime **Description:** `AssignedFrom` represents the date from when the specified steps would be assigned to the associates.
`AssignedUntil`	**Input Type:** System.DateTime **Description:** `AssignedUntil` represents the end date: the time when the task(s) are to be assigned to associates.

Method: SetForwardedTasks	
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to set forwarded tasks.
Return Conditions	
`SetForwardedTasks` returns `TRUE` if the Ultimus engine can set forwarded tasks for the logged on user. `SetForwardedTasks` returns `FALSE` if the Ultimus engine cannot set forwarded tasks for the logged on user.	

SetUserEmailAddress

Method: SetUserEmailAddress	

Method description: This method sets the logged on user's e-mail address.
C# method call:
`public bool SetUserEmailAddress(string strSessionID, string strUserEmail, out string strErr)`
VB.NET method call:
`Public Function SetUserEmailAddress(ByVal strSessionID As String, ByVal strUserEmail As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user; the logged on user should have the necessary rights to perform the current task.
`strUserEmail`	**Input Type:** System.String **Description:** `strUserEmail` specifies the logged on user's e-mail address.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to set the logged on user's e-mail address.

Method: SetUserEmailAddress
Return Conditions
`SetUserEmailAddress` returns `TRUE` if the Ultimus engine can set the logged on user's e-mail address.
`SetUserEmailAddress` returns `FALSE` if the Ultimus engine cannot set the logged on user's e-mail address.

SetUserPreferences

Method: SetUserPreferences	

Method description: This method sets the specified user's preferences.
 	
C# method call:
`public bool SetUserPreferences(string strSessionID, object objUserPreferences, out string strErr)`
 	
VB.NET method call:
`Public Function SetUserPreferences(ByVal strSessionID As String, ByVal objUserPreferences As Object, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user.
`objUserPreferences`	**Input Type:** System.Object **Description:** `objUserPreferences` represents a serialized object of `Ultimus.ClientServices.UserPreferences` class.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to set the user's preferences.
Return Conditions	
`SetUserPreferences` returns `TRUE` if the Ultimus engine can set the user's preferences.	
`SetUserPreferences` returns `FALSE` if the Ultimus engine cannot set the user's preferences.	

ShareViewWithUser

Method: ShareViewWithUser	

Method description: This method shares an identified view in the input parameters with a specified user.
 	
C# method call:
`public bool ShareViewWithUser(string strSessionID, string strViewName, string strViewID, string strSharedUser, out string strErr)`
 	
VB.NET method call:
`Public Function ShareViewWithUser(ByVal strSessionID As String, ByVal strViewName As String, ByVal strViewID As String, ByVal strSharedUser As String, ByRef strErr As String) As Boolean`	

Input	
`SessionID`	**Input Type:** System.String **Description:** `SessionID` represents the logged on user's session ID; the user should have the rights to abort the incident.
`strViewName`	**Input Type:** System.String **Description:** `strViewName` represents the name of the view to be shared.
`strViewID`	**Input Type:** System.String **Description:** `strViewID` represents the View ID that is to be shared.
`strSharedUser`	**Input Type:** System.String **Description:** `strSharedUser` represents the short name of the user with whom the view is to be shared.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine if it attempts to share the view.
Return Conditions	
`ShareViewWithUser` returns `TRUE` if the specified view is shared with the specified user's views. `ShareViewWithUser` returns `FALSE` if the specified view cannot be shared with the specified user's views.	

TakeBackTask

Method: TakeBackTask	

Method description: This method takes back a task previously assigned to another user.
 	
C# method call:
`public bool TakeBackTask(string strSessionID, string strTaskID, out string strErr)`
 	
VB.NET method call:
`Public Function TakeBackTask(ByVal strSessionID As String, ByVal strTaskID As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user; the logged on user should have the necessary rights to perform the current task.
`strTaskID`	**Input Type:** System.String **Description:** `strTaskID` represents the ID of the task that is to be taken back.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to take back the task.
Return Conditions	
`TakeBackTask` returns `TRUE` if the Ultimus engine can take back the task. `TakeBackTask` returns `FALSE` if the Ultimus engine cannot take back the task.	

UnShareViewWithUser

Method: UnShareViewWithUser	

Method description: This method unshares the identified view from a specified user.
 	
C# method call:
`public bool UnshareViewWithUser(string strSessionID, string strViewID, string strSharedUser, out string strErr)`
 	
VB.NET method call:
`Public Function UnshareViewWithUser(ByVal strSessionID As String, ByVal strViewID As String, ByVal strSharedUser As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user; the logged on user should have the necessary rights to perform the current task.
`strViewID`	**Input Type:** System.String **Description:** `strViewID` specifies the ID of the shared view.
`strSharedUser`	**Input Type:** System.String **Description:** `strSharedUser` specifies the short name of the user sharing the view.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to remove the shared view for the specified user.
Return Conditions	
`UnShareViewWithUser` returns `TRUE` if the Ultimus engine can remove the user from the shared view. `UnShareViewWithUser` returns `FALSE` if the Ultimus engine cannot remove the user from the shared view.	

UpdateGroupView

Method: UpdateGroupView	

Method description: This method updates a specified group view with the group view information provided.
 	
C# method call:
`public bool UpdateGroupView(string strSessionID, string strOrgName, string strGroupID, string strViewName, string strViewID, object objViewInfo, out string strErr)`
 	
VB.NET method call:
`Public Function UpdateGroupView(ByVal strSessionID As String, ByVal strOrgName As String, ByVal strGroupID As String, ByVal strViewName As String, ByVal strViewID As String, ByVal objViewInfo As Object, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user; the logged on user should have the necessary rights to perform the current task.
`strOrgName`	**Input Type:** System.String **Description:** `strOrgName` represents the organization name.
`strGroupID`	**Input Type:** System.String **Description:** `strGroupID` represents the group ID of the group whose view is to be updated.
`strViewName`	**Input Type:** System.String **Description:** `strViewName` is the name of the view to be updated.
`strViewID`	**Input Type:** System.String **Description:** `strViewID` is the ID of the specified view.
`objViewInfo`	**Input Type:** System.Object **Description:** `objViewInfo` represents a serialized of the `ViewInfo` class.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to update the group view.

Method: UpdateGroupView
Return Conditions
`UpdateGroupView` returns `TRUE` if the Ultimus engine can update the group view.
`UpdateGroupView` returns `FALSE` if the Ultimus engine cannot update the group view.

UpdateUserView

Method: UpdateUserView	

Method description: This method updates a given view with the view information provided.
C# method call:
`public bool UpdateUserView(string SessionID, string strViewID, string strViewName, object objViewInfo, out string strErr)`
VB.NET method call:
`Public Function UpdateUserView(ByVal strSessionID As String, ByVal strViewID As String, ByVal strViewName As String, ByVal objViewInfo As Object, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user; the logged on user should have the necessary rights to perform the current task.
`strViewID`	**Input Type:** System.String **Description:** `strViewID` represents the ID of the specified view.
`strViewName`	**Input Type:** System.String **Description:** `strViewName` specifies the name of the view to be updated.
`objViewInfo`	**Input Type:** System.Object **Description:** `objViewInfo` represents a serialized object of `Ultimus.ClientServices.ViewInfo` class.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to update the user view.

Method: UpdateUserView
Return Conditions
`UpdateUserView` returns `TRUE` if the Ultimus engine can update the user view.
`UpdateUserView` returns `FALSE` if the Ultimus engine cannot update the user view.

UpdateViewFolder

Method: UpdateViewFolder	

Method description: This method assigns a view to be in a specified folder.
C# method call:
`public bool UpdateViewFolder(string strSessionID, string strViewID, string strFolderID, out string strErr)`
VB.NET method call:
`Public Function UpdateViewFolder(ByVal strSessionID As String, ByVal strViewID As String, ByVal strFolderID As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user; the logged on user should have the necessary rights to perform the current task.
`strViewID`	**Input Type:** System.String **Description:** `strViewID` specifies the ID of the view.
`strFolderID`	**Input Type:** System.String **Description:** `strFolderID` specifies the ID of the folder to be updated.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to update the folder view.
Return Conditions	
`UpdateViewFolder` returns `TRUE` if the Ultimus engine can update the folder for the specified view.	
`UpdateViewFolder` returns `FALSE` if the Ultimus engine cannot update the folder for the specified view.	

UpdateViewFolderName

Method: UpdateViewFolderName	

Method description: This method updates an existing folder's name to be identified with a specified new name.
 	
C# method call:
`public bool UpdateViewFolderName(string strSessionID, string strFolderID, string strNewName, out string strErr)`
 	
VB.NET method call:
`Public Function UpdateViewFolderName(ByVal strSessionID As String, ByVal strFolderID As String, ByVal strNewName As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user.
`strFolderID`	**Input Type:** System.String **Description:** `strFolderID` specifies the ID of the folder in which its name is to be updated.
`strNewName`	**Input Type:** System.String **Description:** `strNewName` specifies the updated name of the folder.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to update the view folder's name.
Return Conditions	
`UpdateViewFolderName` returns `TRUE` if the Ultimus engine can rename the specified folder. `UpdateViewFolderName` returns `FALSE` if the Ultimus engine cannot rename the specified folder.	

UpdateViewSequenceNumberInFolder

Method: UpdateViewSequenceNumberInFolder	

Method description: This method enables the calling application to specify an ordered list of view IDs in which the views contained in a specified folder should be ordered.
 	
C# method call:
`public bool UpdateViewSequenceNumberInFolder(string strSessionID, string strFolderID, string[] strArrViewIDs, out string strErr)`
 	
VB.NET method call:
`Public Function UpdateViewSequenceNumberInFolder(ByVal strSessionID As String, ByVal strFolderID As String, ByVal strArrViewIDs() As String, ByRef strErr As String) As Boolean`	

Input	
`strSessionID`	**Input Type:** System.String **Description:** `strSessionID` represents the session ID of the logged on user.
`strFolderID`	**Input Type:** System.String **Description:** `strFolderID` specifies the ID of the folder.
`strArrViewIDs`	**Input Type:** System.String **Description:** `strArrViewIDs` is a sorted list of view IDs in which the views are to appear in a folder.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to update the view sequence number in a folder.
Return Conditions	
`UpdateViewSequenceNumberInFolder` returns `TRUE` if the Ultimus engine can update the view sequence number in the folder. `UpdateViewSequenceNumberInFolder` returns `FALSE` if the Ultimus engine cannot update the view sequence number in the folder.	

ValidateUser

Method: ValidateUser	

Method description: This method should take a user's credentials and return whether these are correct.
 	
C# method call:
`public bool ValidateUser(string strDomain, string strUser, string strPassword, out string strErr)`
 	
VB.NET method call:
`Public Function ValidateUser(ByVal strDomain As String, ByVal strUser As String, ByVal strPassword As String, ByRef strErr As String) As Boolean`	

Input	
`strDomain`	**Input Type:** System.String **Description:** `strDomain` specifies the name of the domain to which the user belongs.
`strUser`	**Input Type:** System.String **Description:** `strUser` specifies the name of the user.
`strPassword`	**Input Type:** System.String **Description:** `strPassword` specified the user's password.
Output	
`strErr`	**Output Type:** System.String **Description:** `strErr` contains any error message returned by the Ultimus engine when it attempts to validate the user.
Return Conditions	
`ValidateUser` returns `TRUE` if the Ultimus engine can validate the user. `ValidateUser` returns `FALSE` if the Ultimus engine cannot validate the user.	
Comments	
Execution: • If the user name, domain, and password passed represents a valid user, `TRUE` is returned. • If any of the credentials are empty or invalid, `FALSE` is returned. • If there is any exception in the execution of the method, `FALSE` is returned and the exception message is passed as the output parameter.	

6

Process level Web service methods

This section outlines process level Web service methods and how to use them. Ultimus Adaptive BPM Suite provides a variety of Web service methods for launching business processes, completing steps, returning steps, and aborting incidents. Any tool or application that is capable of consuming WSDL 1.1-, SOAP 1.1-, and SOAP 1.2-compliant Web services can leverage these methods.

Refer to the following sections to learn more about process level Web service methods:

- *Overview*
- *Classes used as parameters for Ultimus process level Web service methods*
- *Method signatures for Ultimus process level Web service methods*

Overview

A business process must be enabled to use Web services before the business process can be accessed with Web services. Accessing a business process through a Web service interface has a number of advantages:

- A user can acquire launch information for a business process (a list of element names and initial element values in the Begin step of the process).
- A user can launch an incident of a business process.
- The Web service interface can be used to return all active tasks for a given user associated with one or more incidents of the business process.
- A user can examine task level schema element values within any given individual task.
- A user can set schema element values for a task and submit the task to Ultimus BPM Server such that it is sent to the next step recipient.
- A user can set schema element values for a task and submit the task to Ultimus BPM Server such that it is sent to/returned from the appropriate process recipient.
- A user can abort an active process incident.

To enable a business process to be accessed by Web services, follow these steps:

1. Open Ultimus Process Administrator. Ensure the **Processes** tab is selected.
2. A list of all business processes published to Ultimus BPM Server are listed. Right-click on the business process you want to access through Web services, then select **Enable Web Service**, as shown in Figure 16.

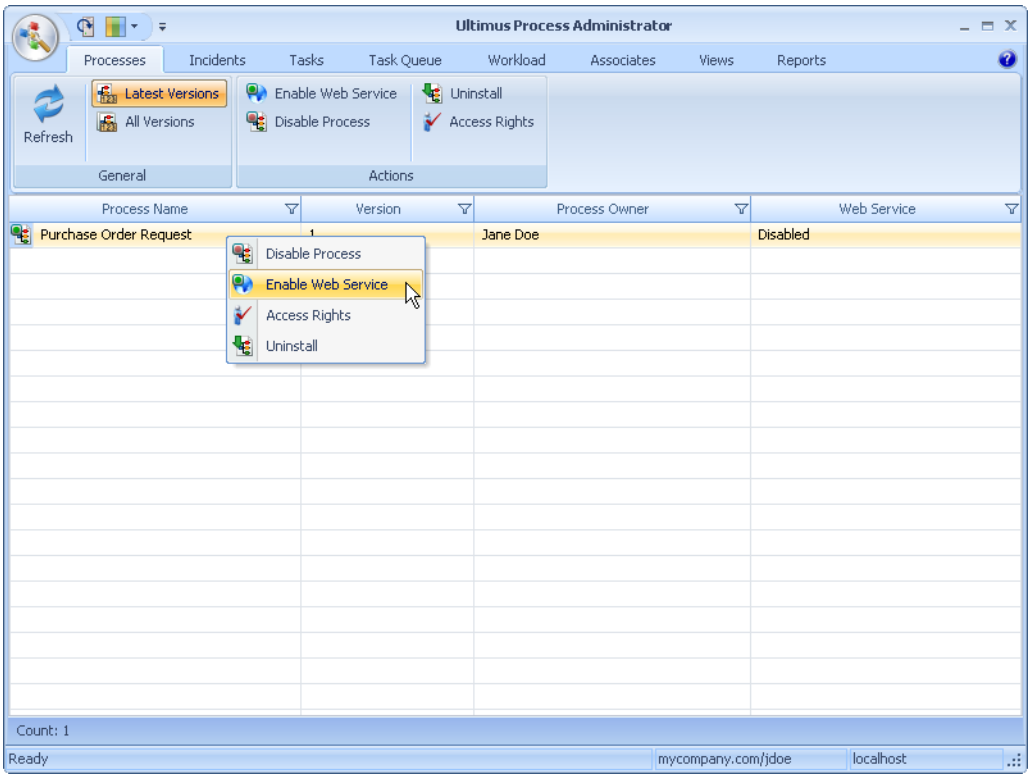

Figure 16. Enabling Web service functionality in Ultimus Process Administrator

The business process can now be accessed through Web services.

Once a Web service for a published business process has been enabled, the following methods for invoking a business process or completing a step can be made:

- *AbortIncident*
- *CompleteStep*
- *GetActiveTasks*
- *GetLaunchInformation*
- *GetTaskInformation*
- *LaunchIncident*
- *ReturnStep*

To view the Web service methods for a business process, follow these guidelines:

1. Load the `ASMX` file for the particular process in Internet Explorer at the following URL: `http://`*`BPMServer`*`/plwebservices/`*`process_name`*`.asmx` (as shown in Figure 17).

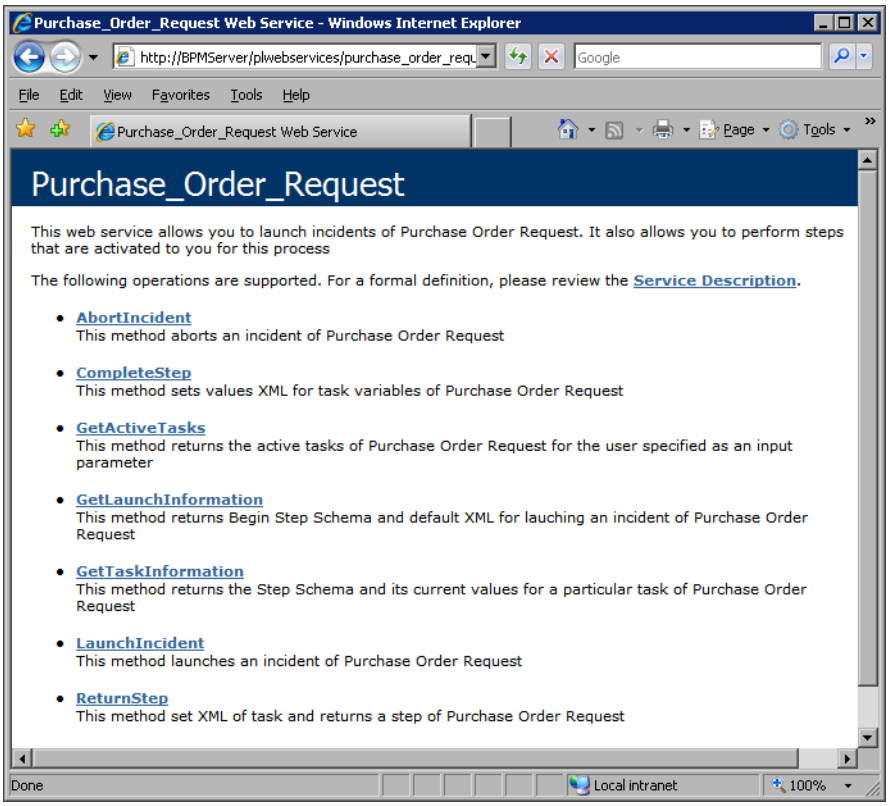

Figure 17. Viewing Web Service methods in Internet Explorer

Note: It is not possible to enable a Web service for a process if (a) the process name contains a period (.), (b) the process name contains a hyphen (−), or (c) the process name start with a numerical character.

Furthermore, if the proces name contains spaces (such as “Purchase Order Request”), when viewing Web service methods in a Web browser, replace any spaces with underscores (as in “Purchase_Order_Request”).

2. To view the XML behind these Web services, either click on the **Service Description** hyperlink of the `ASMX` page or enter the following URL into Internet Explorer's address bar: `http://`*`BPMServer`*`/plwebservices/`*`processname`*`.asmx?WSDL`. The following Web page appears, as shown in Figure 18.

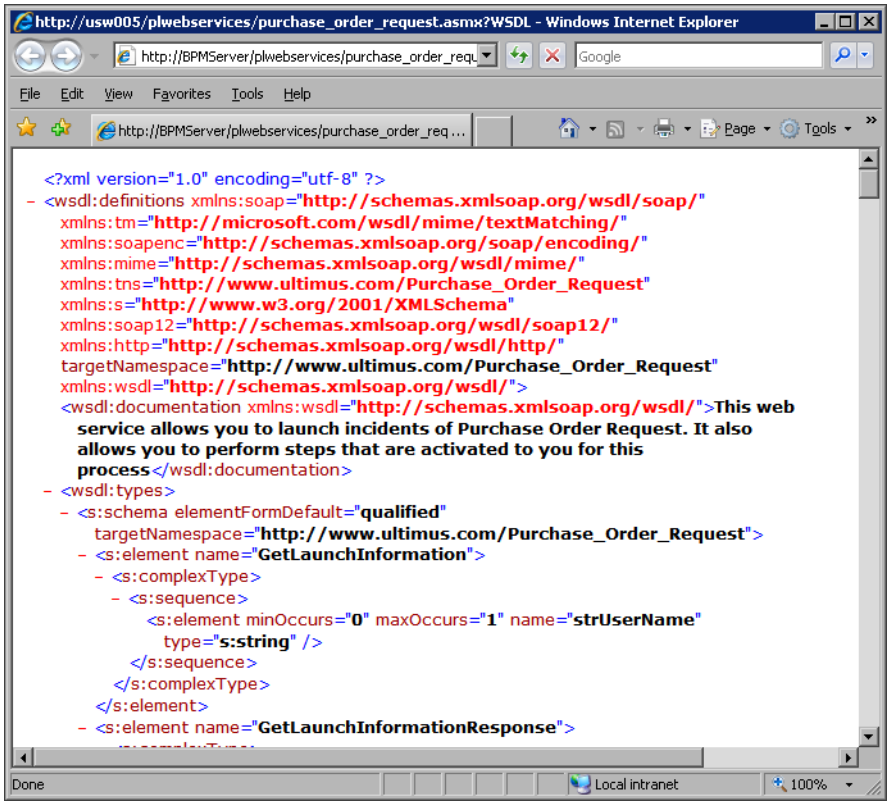

Figure 18. Viewing the XML behind Web services in Internet Explorer

Classes used as parameters for Ultimus process level Web service methods

The following classes are used to output parameters in Web service methods:

- *CTaskInfo*
- *SchemaFile*

CTaskInfo

Class: CTaskInfo	
Class description: The `CTaskInfo` class encapsulates information about a specific task. An array of `CTaskInfo` objects is returned when the `GetActiveTasks` Web service method is triggered.	
Members	
`m_strStepLabel`	**Type:** System.String **Description:** `m_strStepLabel` is the step name within the business process.
`m_strSummary`	**Type:** System.String **Description:** `m_strSummary` is the incident summary for the business process.
`m_nIncidentNum`	**Type:** System.Int32 **Description:** `m_nIncidentNum` is the unique incident number for the process incident.

Class: CTaskInfo

Example

```
// This code snippet is written in C#
// MyProcess is the name of the process against which the Web services have been
enabled.
MyProcess PL = new MyProcess ();
string error;
// Array of type CTaskInfo to contain all tasks information.
CTaskInfo[] ActiveTasks;
try
    {
    // Call method GetActiveTasks() to get information about all tasks.
    if (PL.GetActiveTasks("user name", out ActiveTasks, out error))
    {
        // Loop through all objects of class CTaskInfo and display them in a listview
        for (int i = 0; i < ActiveTasks.Length; i++)
        {
            ListViewItem lItem = new ListViewItem();
            lItem.Text = ActiveTasks[i].m_strStepLabel;
            lItem.SubItems.Add(ActiveTasks[i].m_nIncidentNum.ToString());
            lItem.SubItems.Add(ActiveTasks[i].m_strSummary);
            LstActiveTasks.Items.Add(lItem);
        }
    }
    else
    {
        MessageBox.Show("Failed to get active tasks. Error = " + error);
    }
}
catch (Exception ex)
{
    MessageBox.Show("EXCEPTION:\nMessage = " + ex.Message);
}
```

SchemaFile

Class: SchemaFile	
Class description: The `SchemaFile` class encapsulates Incident and Task XML schema file name and schema data. An array of `SchemaFile` objects is returned when the `GetLaunchInformation` or `GetTaskInformation` Web service method is triggered.	
Members	
`SchemaData`	**Type:** System.String **Description:** `SchemaData` is the actual data contained in the XML schema that exists in an object of `SchemaFile`. `SchemaFile` may be Incident XML schema which are to contain Incident schema data. Alternatively, SchemaFile may contain Task XML schema which are to contain Task schema data.
`SchemaFileName`	**Type:** System.String **Description:** `SchemaFileName` represents the name of the schema file.

Class: SchemaFile

Example

```
// This code snippet is written in C#
// MyProcess is the name of the process for which web services have been enabled.
MyProcess PL = new MyProcess();
// An array of SchemaFile class which contains all schema information returned by
method GetLaunchInformation.
SchemaFile[] schemaFiles;
// Default launch XML
string defaultXML;
string error;
try
{
   if (PL.GetLaunchInformation("User Name", out schemaFiles, out defaultXML, out
   error))
   {
      // Display default XML in a text box.
      TxtDefaultXML.Text = defaultXML;
      // Loop through all schema files and show them in a listview control.
      foreach (ProcessLevelWebServices.SchemaFile sFile in schemaFiles)
      {
         ListViewItem item = new ListViewItem(sFile.SchemaFileName);
         item.SubItems.Add(sFile.SchemaData);
         LstSchemaFilesLaunch.Items.Add(item);
      }
   }
   else
   {
      MessageBox.Show("Failed To Get Launch Information");
   }
}
catch (Exception ex)
{
   MessageBox.Show("EXCEPTION:\nMessage = " + ex.Message);
}
```

Method signatures for Ultimus process level Web service methods

For each of the following Ultmus process level Web service methods, a description of the method, its parameters, comments (where applicable), and an example of each is also included. Outlined below are the method signatures for all Ultimus process level Web service methods:

- *AbortIncident*
- *CompleteStep*
- *GetActiveTasks*
- *GetLaunchInformation*
- *GetTaskInformation*
- *LaunchIncident*
- *ReturnStep*

AbortIncident

Method: AbortIncident	

Method description: This method aborts a running incident of an Ultimus process.
 	
C# method call:
`public System.Boolean AbortIncident (System.String strUserName, System.Int32 nIncidentNumber, System.String strSummary, out System.String strError)`
 	
VB.NET method call:
`Public Function AbortIncident(ByVal strUserName As String, ByVal nIncidentNumber As Integer, ByVal strSummary As String, ByRef strError As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` is the short name (or log on ID) of the user assigned to the process' Begin step.
`nIncidentNumber`	**Input Type:** System.Int32 **Description:** `nIncidentNumber` is the process incident number associated with this workflow task.
`strSummary`	**Input Type:** System.String **Description:** The incident summary for this incident can be set by specifying a value for `strSummary`.

Method: AbortIncident	
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains an error message if the method does not abort a running incident of a business process.
Return Conditions	
`AbortIncident` returns `TRUE` if `AbortIncident` was able to be abort the incident successfully. `AbortIncident` returns `FALSE` if the incident is already complete, the specified user does not exist, the specified incident does not exist, or the incident has already been aborted.	
Example	

```
'This example is written in VB.NET
Dim sample As New SampleWS
Dim strError As String = ""
Dim IncidentNo As Integer = 1

   sample.Credentials = System.Net.CredentialCache.DefaultCredentials
   If sample.AbortIncident("domain_name/demo", IncidentNo, "Incident
      aborted through sample", strError) Then
      MessageBox.Show("Incident has been aborted")
   Else
      MessageBox.Show("Cannot abort incident because of the following
      error " & strError)
   End If
```

CompleteStep

Method: CompleteStep

Method description: This method completes the step and sets the task's XML schema data.

C# method call:

```
public bool CompleteStep(string strUserName, ref int nIncidentNumber, string
strStepName, string strSummary, string strMemo, bool bDisableAbort, int nPriority,
string strXML, bool bValidateXML, out string strError)
```

VB.NET method call:

```
Public Function CompleteStep(ByVal strUserName As String, ByRef nIncidentNumber As
Integer, ByVal strStepName As String, ByVal strSummary As String, ByVal strMemo As
String, ByVal bDisableAbort As Boolean, ByVal nPriority As Integer, ByVal strXML
As String, ByVal bValidateXML As Boolean, ByRef strError As String) As Boolean
```

Method: CompleteStep	
Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` is the short name (or log on ID) of the user assigned to the task.
`nIncidentNumber`	**Input Type:** System.Int32 **Description:** `nIncidentNumber` is the process incident number associated with this workflow task.
`strStepName`	**Input Type:** System.String **Description:** `strStepName` is the name of the step within the Ultimus process.
`strSummary`	**Input Type:** System.String **Description:** `strSummary` correlates with the **Incident Summary** field for the Ultimus process incident.
`strMemo`	**Input Type:** System.String **Description:** `strMemo` correlates with the **Memo** field for the Ultimus process incident.
`bDisableAbort`	**Input Type:** System.Boolean **Description:** `bDisableAbort` specifies whether the incident is abortable or not. Specifying `FALSE` makes the incident abortable, whereas specifying `TRUE` makes the process incident not abortable.
`nPriority`	**Input Type:** System.Int32 **Description:** `nPriority` is the priority of the task.
`strXML`	**Input Type:** System.String **Description:** `strXML` takes the task data in XML format.
`bValidateXML`	**Input Type:** System.Boolean **Description:** `bValidateXML` is a boolean value to specify whether to validate the task data XML or not.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains an error message if the method does not complete the task of an Ultimus process incident.

Method: CompleteStep

Return Conditions

`CompleteStep` returns `TRUE` if `CompleteStep` was able to complete the task.

`CompleteStep` returns `FALSE` if the step name is incorrect, the specified user does not exist, or if the specified incident does not exist.

Example

```
// This code snippet is written in C#
Dim sample As New SampleWS
Dim schema() As SchemaFile = Nothing
Dim strXML As String = ""
Dim strError As String = ""
Dim IncidentNo As Integer = 1
Dim strStepName As String = "StepName"

sample.Credentials = System.Net.CredentialCache.DefaultCredentials
If sample.GetTaskInformation("domain_name/demo", IncidentNo, strStepName, schema,
strXML, strError) Then
'schema element values can be changed by modifying the strXML before submitting it.
   If sample.CompleteStep("domain_name/demo", IncidentNo, strStepName,
      "Process summary set through sample", "memo", False, 9, strXML, False,
      strError) Then MessageBox.Show("Step completed successfully")
   End If
   Else
      MessageBox.Show(strError)
   End If
```

GetActiveTasks

Method: GetActiveTasks	

Method description: This method retrieves the active tasks for a specified user.
 	
C# method call:
`public System.Boolean GetActiveTasks (System.String strUserName, out CTaskInfo[] taskList, out System.String strError)`
 	
VB.NET method call:
`Public Function GetActiveTasks(ByVal strUserName As String, ByRef taskList() As CTaskInfo, ByRef strError As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` is the user short name (or log on ID) of a user that is being queried. The format for `strUserName` should be `domain_name/user_short_name`.
Output	
`taskList`	**Output Type:** instance of *CTaskInfo* class (outlined on page 312) **Description:** `taskList` is an array of `CTaskInfo` objects. Each `CTaskInfo` object describes an active task assigned to a specified user.
`strError`	**Output Type:** System.String **Description:** `strError` is a string that contains an error message if the method does not retrieve the active tasks for a specified user.
Return Conditions	
`GetActiveTasks` returns `TRUE` if `GetActiveTasks` was able to be retrieve active tasks. `GetActiveTasks` returns `FALSE` if there are no active tasks for that process.	
Comments	
Each `CTaskInfo` object returned by this method encapsulates attributes that can be used in calls to the `GetTaskInformation`, `CompleteStep`, and `ReturnStep` Web service methods.	

Method: GetActiveTasks

Example

```
'This sample is written in VB.NET
Dim SampleWS As New BPMServer.SampleWebService
Dim TaskInfo() As BPMServer.CTaskInfo
Dim strError As String

If SampleWS.GetActiveTasks("domain_name/demo", TaskInfo, strError) = True Then
   'GetActiveTasks was successful. Loop through active tasks
   Dim NumTasks As Int32
   Dim x As Int32
   NumTasks = TaskInfo.Length

   'Returned task info could be used to trigger
   'the GetTaskInformation, CompleteStep, and ReturnStep
   'Web service methods.
   For x = 0 To NumTasks - 1
      MsgBox(TaskInfo(x).m_nIncidentNum)
      MsgBox(TaskInfo(x).m_strStepLabel)
      MsgBox(TaskInfo(x).m_strSummary)
   Next x
Else
   'GetActiveTasks was not successful, so return StrError
   MsgBox("GetActiveTasks was not successful, and StrError is: " & strError)
End If
```

GetLaunchInformation

Method: GetLaunchInformation	

Method description: This method returns the Begin step's XML schema and default XML to launch an incident. To show unbounded schema elements in the returned XML File, set the minimum occurrence to 1 for unbounded schema elements.
 	
C# method call:
`public bool GetLaunchInformation(string strUserName, out SchemaFile[] BeginSchema, out string strDefaultXML, out string strError)`
 	
VB.NET method call:
`Public Function GetLaunchInformation(ByVal strUserName As String, ByRef BeginSchema() As SchemaFile, ByRef strDefaultXML As String, ByRef strError As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` is the short name (or log on ID) of the user assigned to the process's Begin step.
Output	
`BeginSchema`	**Output Type:** instance of *SchemaFile* class (outlined on page 314) **Description:** `BeginSchema` returns an array of `SchemaFile` class containing the schema and other dependencies of the Begin step.
strDefaultXML	**Output Type:** System.String **Description:** `strDefaultXML` returns the default XML of the Begin step.
`strError`	**Output Type:** System.String **Description:** `strError` contains an error message if the method could not get the information successfully.
Return Conditions	
`GetLaunchInformation` returns `TRUE` if `GetLaunchInformation` was able to be get the information successfully. `GetLaunchInformation` returns `FALSE` if `GetLaunchInformation` could not get the information successfully or if the specified user does not exist.	
Comments	
If the specified `strUserName` value is not a valid Begin step recipient, the `GetLanchInformation` method fails, the method returns `FALSE`, and `strError` returns with a value of `No task for these parameters`.	

Method: GetLaunchInformation

Example

```
'This sample is written in VB.NET
Dim sample As New SampleWS
Dim schema() As SchemaFile = Nothing
Dim strdefaultXML As String = ""
Dim strError As String = ""

sample.Credentials = System.Net.CredentialCache.DefaultCredentials
If sample.GetLaunchInformation("domain_name/demo", schema,
   strdefaultXML, strError) Then
      MessageBox.Show(strdefaultXML)
Else
   MessageBox.Show(strError)
End If
```

GetTaskInformation

Method: GetTaskInformation	

Method description: This method returns the step XML schema and its current values for a particular task.
 	
C# method call:
`public bool GetTaskInformation(string strUserName, int nIncidentNumber, string strStepName, out SchemaFile[] TaskSchema, out string strTaskXML, out string strError)`
 	
VB.NET method call:
`Public Function GetTaskInformation(ByVal strUserName As String, ByVal nIncidentNumber As Integer, ByVal strStepName As String, ByRef TaskSchema() As SchemaFile, ByRef strTaskXML As String, ByRef strError As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` is the user short name (or log on ID) of the user assigned to the step. The format for `strUserName` should be `domain_name/user_short_name`.
`nIncidentNumber`	**Input Type:** System.Int32 **Description:** `nIncidentNumber` is the process incident number associated with this task.
`strStepName`	**Input Type:** System.String **Description:** `strStepName` is the name of the step within the Ultimus process.
Output	
`TaskSchema`	**Input Type:** instance of *SchemaFile* class (outlined on page 314) **Description:** `TaskSchema` returns an array of `SchemaFile` class containing the schema and other dependencies of the specified step within the business process.
`strTaskXML`	**Output Type:** System.String **Description:** `strTaskXML` returns the default XML of the specified step.
`strError`	**Output Type:** System.String **Description:** `strError` contains an error message if the method does not retrieve task information for a particular active task in an Ultimus process.

Method: GetTaskInformation

Return Conditions

`GetTaskInformation` returns `TRUE` if `GetTaskInformation` was able to retrieve the specified task information.

`GetTaskInformation` returns `FALSE` if the step name is incorrect, the specified user does not exist, or if the specified incident does not exist.

Example

```
'This sample is written in VB.NET
Dim sample As New SampleWS
Dim schemas() As SchemaFile = Nothing
Dim strXML As String = ""
Dim strError As String = ""
Dim IncidentNo As Integer = 1
Dim strStepName As String = "StepName"
Dim schema As SchemaFile

sample.Credentials = System.Net.CredentialCache.DefaultCredentials
If sample.GetTaskInformation("domain_name/demo", IncidentNo, strStepName, schemas,
strXML, strError) Then
   'Step's default XML
   MessageBox.Show(strXML)
      'retrieving step schemas
      For Each schema In schemas
         'Schema File name
         MessageBox.Show(schema.SchemaFileName)
         'Schema data
         MessageBox.Show(schema.SchemaData)
      Next
   Else
      MessageBox.Show("Failed to get the task information because of the following
      error " & strError)
   End If
```

LaunchIncident

Method: LaunchIncident	

Method description: This method launches a process incident.
 	
C# method call:
`public bool LaunchIncident(string strUserName, string strSummary, string strMemo, bool bDisableAbort, int nPriority, string strXML, bool bValidateXML, out int nIncidentNo, out string strError)`
 	
VB.NET method call:
`Public Function LaunchIncident(ByVal strUserName As String, ByVal strSummary As String, ByVal strMemo As String, ByVal bDisableAbort As Boolean, ByVal nPriority As Integer, ByVal strXML As String, ByVal bValidateXML As Boolean, ByRef nIncidentNo As Integer, ByRef strError As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` is the short name (or log on ID) of the user assigned to the process' Begin step.
`strSummary`	**Input Type:** System.String **Description:** The incident summary for this incident can be set by specifying a value for `strSummary`.
`strMemo`	**Input Type:** System.String **Description:** `strMemo` is correlates with the **Memo** field for the process incident.
`bDisableAbort`	**Input Type:** System.Boolean **Description:** `bDisableAbort` specifies whether the incident is abortable or not. Specifying `FALSE` makes the incident abortable, whereas specifying `TRUE` makes the process incident not abortable.
`nPriority`	**Input Type:** System.Int32 **Description:** `nPriority` is the priority of the task.
`strXML`	**Input Type:** System.String **Description:** `strXML` specifies the XML schema of the task.
`bValidateXML`	**Input Type:** System.Boolean **Description:** `bValidateXML` is a boolean value to specify whether to validate the task data XML or not.

Method: LaunchIncident	
Output	
`nIncidentNo`	**Input Type:** System.Int32 **Description:** `nIncidentNo` returns the launched incident number.
`strError`	**Output Type:** System.String **Description:** `strError` contains an error message if the method does not launch a new incident of a business process.
Return Conditions	
`LaunchIncident` returns `TRUE` if `LaunchIncident` was able to launch the incident successfully. `LaunchIncident` returns `FALSE` if `LaunchIncident` could not be launch the incident successfully or if the specified user does not exist.	
Example	

```
'Note: This sample code is written in VB.NET
Dim sample As New SampleWS
Dim schema() As SchemaFile = Nothing
Dim strXML As String = ""
Dim strError As String = ""
Dim IncidentNo As Integer

sample.Credentials = System.Net.CredentialCache.DefaultCredentials
If sample.GetLaunchInformation("domain_name/demo", schema, strXML, strError) Then
   'schema element values can be changed by modifying the strXML before submitting
   it.
      If sample.LaunchIncident("domain_name/demo", "process summary", "memo",
      False, 9, strXML, False, IncidentNo, strError) Then
         MessageBox.Show("New incident " & IncidentNo & " has been launched")
      End If
   Else
      MessageBox.Show(strError)
   End If
```

ReturnStep

Method: ReturnStep	

Method description: This method sets the XML of a task, then return the task.
 	
C# method call:
`public bool ReturnStep(string strUserName, ref int nIncidentNumber, string strStepName, string strSummary, string strMemo, bool bDisableAbort, int nPriority, string strXML, bool bValidateXML, out string strError)`
 	
VB.NET method call:
`Public Function ReturnStep(ByVal strUserName As String, ByRef nIncidentNumber As Integer, ByVal strStepName As String, ByVal strSummary As String, ByVal strMemo As String, ByVal bDisableAbort As Boolean, ByVal nPriority As Integer, ByVal strXML As String, ByVal bValidateXML As Boolean, ByRef strError As String) As Boolean`	

Input	
`strUserName`	**Input Type:** System.String **Description:** `strUserName` is the short name (or log on ID) of the user assigned to the task.
`nIncidentNumber`	**Input Type:** System.Int32 **Description:** `nIncidentNumber` is the process incident number associated with this task.
`strStepName`	**Input Type:** System.String **Description:** `strStepName` is the name of the step within the Ultimus process.
`strSummary`	**Input Type:** System.String **Description:** The incident summary for this incident can be set by specifying a value for `strSummary`.
`strMemo`	**Input Type:** System.String **Description:** `strMemo` correlates with the **Memo** field for the Ultimus process incident.
`bDisableAbort`	**Input Type:** System.Boolean **Description:** `bDisableAbort` specifies whether the incident is abortable or not. Specifying `FALSE` makes the incident abortable, whereas specifying `TRUE` makes the process incident not abortable.
`nPriority`	**Input Type:** System.Int32 **Description:** `nPriority` is the priority of the task.

Method: ReturnStep	
`strXML`	**Input Type:** System.String **Description:** `strXML` specifies the XML of the task.
`bValidateXML`	**Input Type:** System.Boolean **Description:** `bValidateXML` is a boolean value which specifies whether to validate the task data or not.
Output	
`strError`	**Output Type:** System.String **Description:** `strError` contains an error message if the method does not return the active step back to the previous user.

Return Conditions

`ReturnStep` returns `TRUE` if `ReturnStep` was able to execute successfully.

`ReturnStep` returns `FALSE` if the step name is incorrect, the specified user does not exist, or if the incident does not exist.

Example

```
// This code snippet is written in C#
Dim sample As New SampleWS
Dim schema() As SchemaFile = Nothing
Dim strXML As String = ""
Dim strError As String = ""
Dim IncidentNo As Integer = 1
Dim strStepName As String = "StepName"

sample.Credentials = System.Net.CredentialCache.DefaultCredentials
If sample.GetTaskInformation("domain_name/demo", IncidentNo, strStepName, schema,
strXML, strError) Then
    'schema element values can be changed by modifying the strXML before submitting
    it.
        If sample.ReturnStep("domain_name/demo", IncidentNo, strStepName, "returned
        step", "memo", False, 9, strXML, False, strError) Then
            MessageBox.Show("Step returned successfully")
        End If
    Else
        MessageBox.Show("Unable to return step because of the following error " &
        strError)
End If
```

7

Initiating business process incidents from third-party applications

This section outlines how to initiate business process incidents from third-party applications. The ability to initiate a business process incident from a third-party application is one of the most basic (yet useful) forms of integration the Ultimus EIK offers. Any application capable of sending an e-mail message, generating a text file, or delivering a Web service can start a business process.

The phrase "initiating the business process" refers to when a third-party application can complete the first (or Begin step) of a business process. With the techniques outlined in this section, any third-party application can complete the Begin step. This means that the third-party application not only initiates the business process, but also provides the data necessary to complete the Begin step.

Here are two examples of how this function might be used:

- An accounting software program determines that an accounts receivable from a customer is past due. It can use this capability to initiate a process for collection. The accounting software can pass the name of the customer, the address, the amount due, and other relevant information to the process. This process can then send an e-mail message, followed by a letter, to the customer. It can then wait for a response, then escalate the receivable to different levels if the issue is not resolved in a timely fashion.
- A document imaging software package must route new images to various individuals for approval as a part of an automobile insurance claims process. Incoming images are scanned at a scanning station and relevant information is entered. The scanning station saves the images in a database and creates an index number to identify each image. The scanning station then uses the text file launch capability to initiate the "Claims" process and send the data and index number to the business process that handles the routing.

This section discusses the following ways to initiate business process incidents from third-party applications:

- *Initiating business process incidents using text command files*
- *Initiating a business process incident via e-mail*

Initiating business process incidents using text command files

A business process can be initiated by generating a formatted ASCII text file and saving it on the Ultimus BPM Server. This allows any third-party application that can generate an ASCII text file to initiate a process incident. How to launch business processes using text command files, the format of such text files, and examples of how they can be used are described below:

1. Create an ASCII text file in the Ultimus BPM Server's `Common` directory. The file's name should be `TXTCMD.???` (where `???` is replaced with any three letters or numbers).
2. Format the file using the exact specifications provided below, *Format specifications for text command files*.
3. Save, then close the text file. As an example, refer to *An example of initiating a business process using a text command file* section.

Format specifications for text command files

The format of the text command file must be exactly as shown below. The following are syntax variables that can be used in the formatted ASCII text file:

- **ElementRef:** This references an element name for the Begin step of the business process.
- **Owner:** This references the owner of the Begin step. The specified owner in the text command file is named the initiator of the process incident when the process is launched. Moreover, the specified owner must be a valid Begin step user for the published business process.

Formatting the ASCII text file must follow the specifications detailed in the following table.

Table 26. Formatting requirements for an ASCII text file to launch a business process

Line Number of the text file	Required Content	Example
Line 1	The name of the business process to be initiated	Purchase Order
Line 2	`TaskData.Global.MyGlobalElement =data`	`TaskData.Global.MyGlobal Element=Additional Clients`
Line *X* (*X* being a line of text after line 3)	`TaskData.Global.MyGlobalElement =New Hardware`	`TaskData.Global.MyOther GlobalElement=New Hardware`
Line *X*+1	Owner=domain name/owner's short name	Owner=YourDomain/Ksmith

Line 1 is required. The remaining lines are optional, based on the process design and what information is needed to complete the Begin step. If the format of the ASCII text file does not meet the required specifications, then either incorrect values are passed to Ultimus BPM Server or Ultimus BPM Server fails to process the text command line.

An example of initiating a business process using a text command file

Below is a process map to solicit purchases. In this example, the business process begins when an initiator launches the business process from Ultimus Client by submitting the Begin step. (In this scenario, the Begin step is the step labeled "Request.")

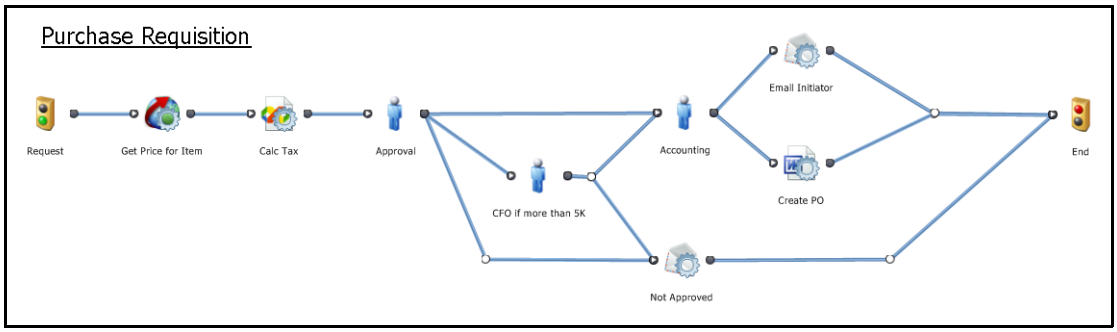

Figure 19. Process map which describes a business process to solicit purchases

When initiated, the Purchase Order form for the Begin step appears (as shown in Figure 20), and the initiator enters in the pertinent data.

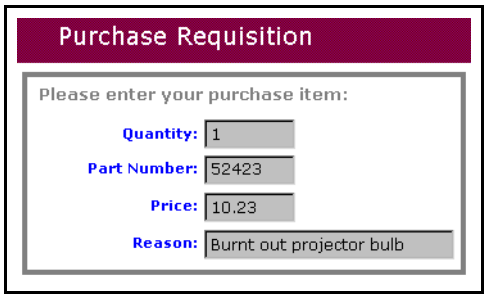

Figure 20. Purchase Order form for the Begin step

As the form is updated, the data is transferred to the step schema for the Begin step (as shown in Figure 21). When the form is sent, the data is transferred from the step schema to the global schema.

Figure 21. Step schema for the Begin step sent to the global schema

In order to initiate this business process from a text command file, create a text file named `TXTCMD.001` in the format shown below, then place it in the `Common` directory (which is within Ultimus Adaptive BPM Suite's installation directory).

```
Purchase Requisition
TaskData.Global.Quantity=1
TaskData.Global.PartNumber=52423
TaskData.Global.Price=10.23
TaskData.Global.Reason=Burnt out projector bulb
Owner=YourDomain/Ksmith
```

In this example, the e-mail instructs Ultimus BPM Server to launch the `Purchase Requisition`, where `Purchase Requisition` is the name as was given to the business process in Ultimus BPM Studio before it is published. It then instructs Ultimus BPM Server to populate the form for the first step with data. The XML schema element name is specified first, then the value. It also instructs Ultimus BPM Server to assign the initiator of the process as `Kathy Smith` (by following `domain name/user name` format). Ultimus BPM Server launches this business process automatically and then deletes the e-mail message.

Note: Data values for unbounded schema elements can be set assuming the specific elements of the unbounded schema element are initialized in the business process. For example, if the name of the unbounded schema element is `PartStatus` (having three elements), the values for `PartStatus` could be initialized using the following format:

```
TaskData.Global.PartStatus[0]=value_1
TaskData.Global.PartStatus[1]=value_2
TaskData.Global.PartStatus[2]=value_3
```

The Ultimus BPM service on the computer hosting Ultimus BPM Server reads the text file and creates a new incident of the process named "Purchase Requisition." It automatically completes the first step using the data in the text file.

Finally, the text file is deleted from the `Common` directory.

Initiating a business process incident via e-mail

A business process may also be initiated simply by sending a formatted e-mail message to the Ultimus BPM Server. To use this feature, the following conditions must be met:

- Either MAPI or POP3/SMTP e-mail protocol is required.
- E-mail server software must be running.
- An e-mail address must be dedicated for the target Ultimus BPM Server.
- In Ultimus System Administrator, the **Enable Auto-Launch** option must be selected, and Ultimus System Administrator must be configured with the appropriate e-mail information.

The subject of the e-mail must be `ULTIMUS-LAUNCH` and the body of the message must conform to the format specified in the *Initiating business process incidents using text command files* section.

In order to initiate the business process "Purchase Requisition" via e-mail, compose the formatted message, then send it to the Ultimus BPM Server's Auto Launch e-mail address. Note that the e-mail must be a plain text message. Rich text format (RTF) and HTML do not work properly.

An example is shown in Figure 22.

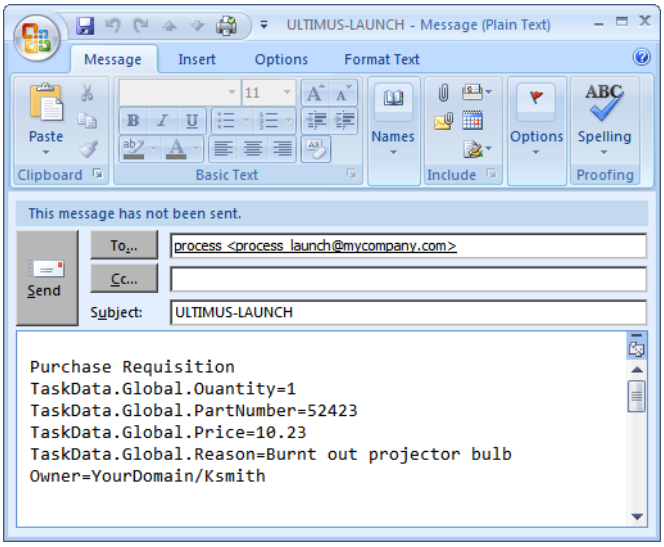

Figure 22. Launching a business process via e-mail

Note: Data values for unbounded schema elements can be set assuming the specific elements of the unbounded schema element are initialized in the business process. For example, if the name of the unbounded schema element is `PartStatus` (having three elements), the values for `PartStatus` could be initialized using the following format:

```
TaskData.Global.PartStatus[0]=value_1
TaskData.Global.PartStatus[1]=value_2
TaskData.Global.PartStatus[2]=value_3
```

E-mail process initiation may be used with either MAPI (Exchange) or Internet (SMTP) e-mail accounts. If used with Microsoft Exchange Server, Ultimus BPM Server must be assigned an Exchange user profile. Once a profile has been assigned, follow these steps to place the Exchange user profile into Ultimus System Administrator:

1. Go to **Start»Programs»Ultimus Adaptive BPM Suite 8.3»Ultimus System Administrator**.
2. Open the properties for the Ultimus BPM Server, then select the **Mail** tab.
3. Select the **Enable Auto Launch** option, then enter the name of the e-mail server and the e-mail account information.

The Ultimus BPM service logs into the designated e-mail account (based on the Auto Launch settings), checks for e-mails titled "Ultimus-Launch," then converts these e-mails to text command files. The Ultimus BPM service then consumes these text command files and generates new incidents for the identified business process.

For additional information, refer to *Configuring the E-Mail Settings node properties* section in *Ultimus System Administrator Help*.

8

The Ultimus Custom OC interface

This section discusses how to use a custom OC interface with Ultimus Organization Charts. At the heart of the Ultimus Adaptive BPM Suite, the Ultimus Organization Charts module is used to maintain departmental and user information. This information is vital to the Ultimus engine's ability to route tasks to approprite process participants. For organizations that already use a third-party application to manage their organizational structure, it can be cumbersome to maintain duplicate information in Ultimus Organization Charts. The Ultimus Custom OC interface can synchronize an existing third-party application with Ultimus Organization Charts. The Custom OC interface allows Ultimus Adaptive BPM Suite to retrieve organizational information from a third-party database or application.

This section discusses the following aspects of the Ultimus Custom OC interface:

- *Overview*
- *Characteristics of the Ultimus Custom OC interface*
- *Minimum requirements of a third-party custom database*
- `SharedInterfaceLib.IAuthentication` *interface methods*
- `SharedInterfaceLib.IOrgChart` *interface methods*
- *Installing the Ultimus Custom OC interface*

Overview

To synchronize Ultimus Adaptive BPM Suite with a third-party application or database, a custom COM component or a custom .NET assembly is required that implements two custom OC interfaces:

- `SharedInterfaceLib.IOrgChart`
- `SharedInterfaceLib.IAuthentication`

This component or assembly works as a bridge between any third-party application and Ultimus Adaptive BPM Suite, as illustrated in Figure 23.

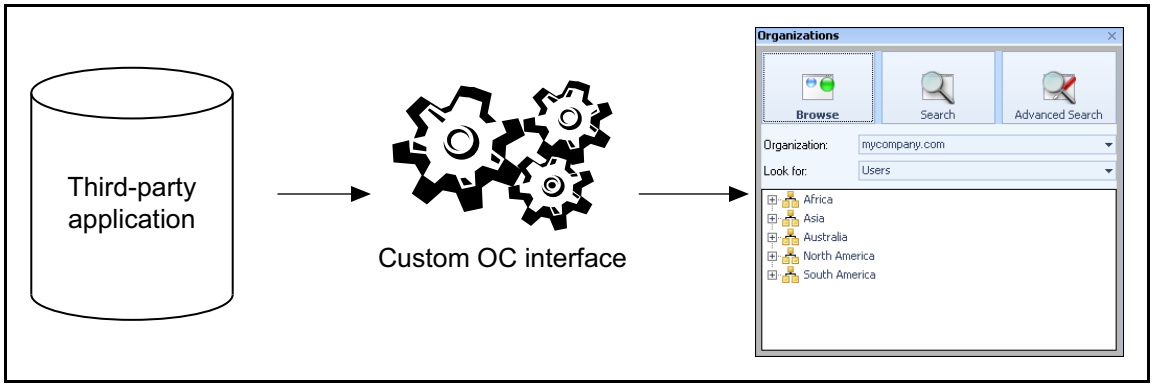

Figure 23. The role the Custom OC interface plays between third-party custom applications and Ultimus Adaptive BPM Suite

Characteristics of the Ultimus Custom OC interface

The Custom OC interface is designed to allow Ultimus Adaptive BPM Suite to communicate with a data source that provides information about Ultimus Adaptive BPM Suite users. This data source can be any file type or format since the COM component can be customized to communicate with that data source.

The Custom OC interface includes two interfaces: `IOrgChart` and `IAuthentication`. Each interface is described briefly below:

- **`IAuthentication`:** The `IAuthentication` interface represents the mechanism to authenticate users from a custom directory. Methods for the `IAuthentication` interface are outlined in *`SharedInterfaceLib.IAuthentication` interface methods* section.
- **`IOrgChart`:** The `IOrgChart` interface represents the mechanism to use third-party applications as organization charts. This interface provides methods which must be used to provide a complete implementation of Custom OC. Methods for the `IOrgChart` interface are outlined in *`SharedInterfaceLib.IOrgChart` interface methods* section.

The external data source completely replaces the Ultimus Organization Charts database tables. User, Department, Manager, Supervisor, and Group data is read from the external data source and presented according to the rules set in the custom application. This custom application reads from the external data source, then passes the identified data through the Custom OC interface to Ultimus Adaptive BPM Suite.

There are limitations to using the Direct Read mode. These are described below.

- In order for custom Ultimus Organization Charts (OC) information to be displayed accurately (as well as tasks routed correctly), each user in the custom OC database structure must have a corresponding supervisor ID and manager ID. A "supervisor" is the individual in a business chart to whom a user directly reports. A "manager" is the top-most supervisor within the department.
- Job Function Groups are not supported by the Custom OC interface.
- When the Custom OC interface is implemented, access rights functionality is not available.

Minimum requirements of a third-party custom database

Despite that the custom database can contain any number of tables and be of any format, there are some minimum requirements for a third-party custom database in order for the Ultimus Custom OC interface to properly function. The following are the minimum requirements for a third-party custom database:

- Each user used in the third-party custom database must have a unique full name. If each full name is not unique, then the Ultimus Custom OC interface could possibly return an incorrect short name value.
- Each user used in the third-party custom database must have a unique short name. If each short name is not unique, then any of the Ultimus Custom OC interface methods which take a short name as an input parameter could possibly return an incorrect value.
- All IDs in each of the tables of the third-party custom database must be unique. If the same IDs exist inside any one table, then any of the Ultimus Custom OC interface methods which take an ID as an input parameter could possibly return an incorrect value.

`SharedInterfaceLib.IAuthentication` interface methods

The `SharedInterfaceLib.IAuthentication` interface represents the mechanism to authenticate users from a custom directory. It is implemented along with `SharedInterfaceLib.IOrgChart` and can be used to verify users from `IOrgChart`. Outlined below are the custom method calls for the `SharedInterfaceLib.IAuthentication` interface (in alphabetical order):

- *Configure*
- *ValidateUser*
- *ValidateUserAndGetSignature*

Configure

Method: Configure	

Method description: This method is called internally to initialize the authentication component.
C# method call:
`void Configure(string bstrProgID)`
VB.NET method call:
`Sub Configure(ByVal bstrProgID As String)`	

Input	
`bstrProgID`	**Input Type:** System.String **Description:** `bstrProgID` represents the program ID

ValidateUser

Method: ValidateUser	

Method description: This method validates a user.
C# method call:
`int ValidateUser(string bstrDomainName, string bstrUserName, string bstrPassword)`
VB.NET method call:
`Function ValidateUser(ByVal bstrDomainName As String, ByVal bstrUserName As String, ByVal bstrPassword As String) As Integer`	

Input	
`bstrDomainName`	**Input Type:** System.String **Description:** `bstrDomainName` represents the user's domain name.
`bstrUserName`	**Input Type:** System.String **Description:** `bstrUserName` represents the user's name.
`bstrPassword`	**Input Type:** System.String **Description:** `bstrPassword` represents the user's password.
Return Conditions	
`ValidateUser` returns `1` if the user is successfully validated. `ValidateUser` returns `0` if the user is not successfully validated.	

ValidateUserAndGetSignature

Method: ValidateUserAndGetSignature	

Method description: This method validates and returns the signature image for that user. Note that this method is not needed to be implemented in the custom authentication as signatures are stored against business organization users and can only be retrieved from the business organization.
 	
C# method call:
`void ValidateUserAndGetSignature(string bstrDomainName, string bstrUserName, string bstrPassword, out object pvtSignature)`
 	
VB.NET method call:
`Sub ValidateUserAndGetSignature(ByVal bstrDomainName As String, ByVal bstrUserName As String, ByVal bstrPassword As String, ByRef pvtSignature As Object)`	

Input	
`bstrDomainName`	**Input Type:** System.String **Description:** `bstrDomainName` represents the user's domain name.
`bstrUserName`	**Input Type:** System.String **Description:** `bstrUserName` represents the user's name.
`bstrPassword`	**Input Type:** System.String **Description:** `bstrPassword` represents the user's password.
Output	
`pvtSignature`	**Output Type:** System.Object **Description:** `pvtSignature` returns the signature image if the log on was successful.

`SharedInterfaceLib.IOrgChart` interface methods

The `SharedInterfaceLib.IOrgChart` interface represents the mechanism to use third-party applications as organization charts. This interface provides methods which must be used to provide a complete implementation of Custom OC. For better readability, related methods are grouped together. Outlined below are the custom method calls for the `SharedInterfaceLib.IOrgChart` interface:

- *"Department" methods in the `IOrgChart` interface*
- *"Group" methods in the `IOrgChart` interface*
- *"Job function" methods in the `IOrgChart` interface*
- *Miscellaneous methods in the `IOrgChart` interface*

- *"Superior" methods in the IOrgChart interface*
- *"User name" methods in the IOrgChart interface*
- *"User information" methods in the IOrgChart interface*

"Department" methods in the IOrgChart interface

Outlined below are the "department" methods in the IOrgChart interface. These methods are listed in alphabetical order:

- *DepartmentIdToName*
- *DoesDepartmentExist*
- *GetDepartmentID*
- *GetDepartmentMembers*
- *GetDepartments*
- *GetParentDepartmentName*
- *IsDepartmentUser*

DepartmentIdToName

Method: DepartmentIdToName	

Method description: This method returns a department's name based on a specified department ID.
C# method call:
`void DepartmentIdToName(string bstrOrgName, string bstrDeptID, out string bstrDeptName)`
VB.NET method call:
`Sub DepartmentIdToName(ByVal bstrOrgName As String, ByVal bstrDeptID As String, ByRef bstrDeptName As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization's name.
`bstrDeptID`	**Input Type:** System.String **Description:** `bstrDeptID` represents the department's ID.
Output	
`bstrDeptName`	**Output Type:** System.String **Description:** `bstrDeptName` represents the department's name.

DoesDepartmentExist

Method: DoesDepartmentExist	

Method description: This method checks for the existence of a specified department.
 	
C# method call:
`int DoesDepartmentExist(string bstrOrgName, string bstrDeptName)`
 	
VB.NET method call:
`Function DoesDepartmentExist(ByVal bstrOrgName As String, ByVal bstrDeptName As String) As Integer`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization's name.
`bstrDeptName`	**Input Type:** System.String **Description:** `bstrDeptName` represents the name of the department.
Return Conditions	
`DoesDepartmentExist` returns `1` if the department is found. `DoesDepartmentExist` returns `0` if the department is not found.	

GetDepartmentID

Method: GetDepartmentID	

Method description: This method returns a department's ID based on a specified department name.
C# method call:
`void GetDepartmentID(string bstrOrgName, string deptName, out string deptID)`
VB.NET method call:
`Sub GetDepartmentID(ByVal bstrOrgName As String, ByVal deptName As String, ByRef deptID As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization's name.
`deptName`	**Input Type:** System.String **Description:** `deptName` represents the department's name.
Output	
`deptID`	**Output Type:** System.String **Description:** `deptID` represents the specified department's ID.

GetDepartmentMembers

Method: GetDepartmentMembers	

Method description: This method retrieves the members of a specified department.
 	
C# method call:
`void GetDepartmentMembers(string bstrOrgName, string deptID, out object UserNames, out object FullNames, out object JobFunctions, out object UserIDs)`
 	
VB.NET method call:
`Sub GetDepartmentMembers(ByVal bstrOrgName As String, ByVal deptID As String, ByRef UserNames As Object, ByRef FullNames As Object, ByRef JobFunctions As Object, ByRef UserIDs As Object)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization's name.
`deptID`	**Input Type:** System.String **Description:** `deptID` represents the department ID.
Output	
`UserNames`	**Output Type:** System.Object **Description:** `UserNames` represents the members' user names within the specified department.
`FullNames`	**Output Type:** System.Object **Description:** `FullNames` represents the members' full names within the specified department.
`JobFunction`	**Output Type:** System.Object **Description:** `JobFunction` represents the members' job functions within the specified department.
`UserID`	**Output Type:** System.Object **Description:** `UserID` represents the members' user IDs within the specified department.

GetDepartments

Method: GetDepartments	

Method description: This method retrieves the subdepartments within a specified department of an organization.
 	
C# method call:
`void GetDepartments(string bstrOrgName, string deptID, out object DeptNames, out object DeptIDs)`
 	
VB.NET method call:
`Sub GetDepartments(ByVal bstrOrgName As String, ByVal deptID As String, ByRef DeptNames As Object, ByRef DeptIDs As Object)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization's name.
`deptID`	**Input Type:** System.String **Description:** `deptID` represents the department ID.
Output	
`DeptNames`	**Output Type:** System.Object **Description:** `DeptNames` represents an array of department names.
`DeptIDs`	**Output Type:** System.Object **Description:** `DeptIDs` represents an array of department IDs.

GetParentDepartmentName

Method: GetParentDepartmentName	

Method description: This method retrieves the name of the parent department.
 	
C# method call:
`void GetParentDepartmentName(string bstrOrgName, string bstrDeptName, out string bstrParentDeptName)`
 	
VB.NET method call:
`Sub GetParentDepartmentName(ByVal bstrOrgName As String, ByVal bstrDeptName As String, ByRef bstrParentDeptName As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`bstrDeptName`	**Input Type:** System.String **Description:** `bstrDeptName` represents the department's name.
Output	
`bstrParentDeptName`	**Output Type:** System.String **Description:** `bstrParentDeptName` represents the parent department's name.

IsDepartmentUser

Method: IsDepartmentUser	

Method description: This method verifies whether a specified user is within a specified department.
 	
C# method call:
`int IsDepartmentUser(string bstrOrgName, string deptID, string deptName, string userName)`
 	
VB.NET method call:
`Function IsDepartmentUser(ByVal bstrOrgName As String, ByVal deptID As String, ByVal deptName As String, ByVal userName As String) As Integer`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization's name.
`deptID`	**Input Type:** System.String **Description:** `deptID` represents the department ID.
`deptName`	**Input Type:** System.String **Description:** `deptName` represents the department's name.
`userName`	**Input Type:** System.String **Description:** `userName` represents the name of the user. `userName` is checked against the department name. If the department name is not provided, then searching is based on the provided department ID.
Return Conditions	
`IsDepartmentUser` returns `1` if the user is found. `IsDepartmentUser` returns `0` if the user is not found.	

"Group" methods in the `IOrgChart` interface

Outlined below are the "group" methods in the `IOrgChart` interface. These methods are listed in alphabetical order:

- *DoesGroupExist*
- *GetGroupID*
- *GetGroupMembers*
- *GetGroups*
- *DoesGroupExist*

- *GetGroupID*
- *GetGroupMembers*
- *GetGroups*
- *GetMemberWeight*
- *GetNextMember*
- *GetWeightedGroupMembers*
- *IsMemberOfGroup*
- *IsWeightedGroup*

DoesGroupExist

Method: DoesGroupExist	

Method description: This method verifies for the existence of a group within a specified organization.
C# method call:
`int DoesGroupExist(string bstrOrgName, string grpName)`
VB.NET method call:
`Function DoesGroupExist(ByVal bstrOrgName As String, ByVal grpName As String) As Integer`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization's name.
`grpName`	**Input Type:** System.String **Description:** `grpName` represents the name of the group whose existence is to be verified.
Return Conditions	
`DoesGroupExist` returns `1` if the group exists. `DoesGroupExist` returns `0` if the group does not exist.	

GetGroupID

Method: GetGroupID	

Method description: This method retrieves the Group ID of a specified group.
 	
C# method call:
`void GetGroupID(string bstrOrgName, string bstrGrpName, out string nGID)`
 	
VB.NET method call:
`Sub GetGroupID(ByVal bstrOrgName As String, ByVal bstrGrpName As String, ByRef nGID As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`bstrGrpName`	**InputType:** System.String **Description:** `bstrGrpName` represents the name of the group.
Output	
`nGID`	**Output Type:** System.String **Description:** `nGID` represents the group ID of the specified group.

GetGroupMembers

Method: GetGroupMembers	

Method description: This method retrieves the members of a specified group. Note that GetGroupMembers works recursively and returns members of subgroups.
 	
C# method call:
`void GetGroupMembers(string bstrOrgName, string grpName, int bFullNames, out object UserNames, out object FullNames)`
 	
VB.NET method call:
`Sub GetGroupMembers(ByVal bstrOrgName As String, ByVal grpName As String, ByVal bFullNames As Integer, ByRef UserNames As Object, ByRef FullNames As Object)`	

Input	
bstrOrgName	**Input Type:** System.String **Description:** bstrOrgName represents the organization name.
grpName	**InputType:** System.String **Description:** grpName represents the name of the group whose members are to be retrieved.
bFullNames	**InputType:** System.Int32 **Description:** bFullNames is a flag if the full name is required or not. If bFullNames is TRUE (non-zero), then the full name is returned in the string array; otherwise, user names are returned in the string array.
Output	
UserNames	**Output Type:** System.Object **Description:** UserNames represents an array of user names in an object.
FullNames	**Output Type:** System.Object **Description:** FullNames represents an array of full names in an object.

GetGroups

Method: GetGroups	

Method description: This method retrieves the names of all groups within a specified organization.
C# method call:
`void GetGroups(string bstrOrgName, out object Groups)`
VB.NET method call:
`Sub GetGroups(ByVal bstrOrgName As String, ByRef Groups As Object)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
Output	
`Groups`	**Output Type:** System.Object **Description:** `Groups` represents an array of group names within the organization.

GetMemberWeight

Method: GetMemberWeight	

Method description: This method retrieves the weight of a specified member within a Weighted group.
C# method call:
`void GetMemberWeight(string bstrOrgName, string grpName, string member, out int weight)`
VB.NET method call:
`Sub GetMemberWeight(ByVal bstrOrgName As String, ByVal grpName As String, ByVal member As String, ByRef weight As Integer)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`grpName`	**Input Type:** System.String **Description:** `grpName` represents the name of the group to which the user belongs.

Method: GetMemberWeight	
member	**Input Type:** System.String **Description:** member represents group member's user name.
Output	
weight	**Output Type:** System.Int32 **Description:** weight represents the weight of the specified group member.

GetNextMember

Method: GetNextMember	

Method description: This method retrieves the next member within a specified group based on the previous member's user name.
C# method call:
`void GetNextMember(string bstrOrgName, string grpName, string prevMember, out string member)`
VB.NET method call:
`Sub GetNextMember(ByVal bstrOrgName As String, ByVal grpName As String, ByVal prevMember As String, ByRef member As String)`	

Input	
bstrOrgName	**Input Type:** System.String **Description:** bstrOrgName represents the organization name.
grpName	**Input Type:** System.String **Description:** grpName represents the name of the group.
prevMember	**Input Type:** System.String **Description:** grpName represents the name of the previous member in the specified group.
Output	
member	**Output Type:** System.String **Description:** member represents the name of the next member within the group.

GetWeightedGroupMembers

Method: GetWeightedGroupMembers	

Method description: This method retrieves the members of a Weighted group, as well as each member's corresponding weight within the group.
 	
C# method call:
`void GetWeightedGroupMembers(string bstrOrgName, string grpName, out object varUserNames, out object varWeights)`
 	
VB.NET method call:
`Sub GetWeightedGroupMembers(ByVal bstrOrgName As String, ByVal grpName As String, ByRef varUserNames As Object, ByRef varWeights As Object)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`grpName`	**Input Type:** System.String **Description:** `grpName` represents the name of the group.
Output	
`varUserName`	**Output Type:** System.Object **Description:** `varUserName` represents an array of user names in an object.
`varWeights`	**Output Type:** System.Object **Description:** `varWeights` represents an array of corresponding user weights in an object.

IsMemberOfGroup

Method: IsMemberOfGroup	

Method description: This method verifies whether a specified user is a member within a specified group.
 	
C# method call:
`int IsMemberOfGroup(string bstrOrgName, string grpName, string userName)`
 	
VB.NET method call:
`Function IsMemberOfGroup(ByVal bstrOrgName As String, ByVal grpName As String, ByVal userName As String) As Integer`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`grpName`	**Input Type:** System.String **Description:** `grpName` represents the name of the group.
`userName`	**Input Type:** System.String **Description:** `userName` represents the user to be verified as a member of the specified group.
Return Conditions	
`IsMemberOfGroup` returns `1` if the user is a member of the group. `IsMemberOfGroup` returns `0` if the user is not a member of the group.	

IsWeightedGroup

Method: IsWeightedGroup	

Method description: This method verifies whether a specified group is a Weighted group.
 	
C# method call:
`int IsWeightedGroup(string bstrOrgName, string grpName)`
 	
VB.NET method call:
`Function IsWeightedGroup(ByVal bstrOrgName As String, ByVal grpName As String) As Integer`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`grpName`	**Input Type:** System.String **Description:** `grpName` represents the name of the group.
Return Conditions	
`IsWeightedGroup` returns `1` if the group is a Weighted group. `IsWeightedGroup` returns `0` if the group is not a Weighted group.	

"Job function" methods in the `IOrgChart` interface

Outlined below are the "job function" methods in the `IOrgChart` interface. These methods are listed in alphabetical order:

- *GetAllJobFunctionsOfPerson*
- *GetFullJobFunctionToID*
- *GetFullJobFunctionToName*
- *GetPrimaryJobFunction*
- *GetUserJobFunctionInDepartment*
- *IDToJobFunction*
- *JobFunctionToName*

GetAllJobFunctionsOfPerson

Method: GetAllJobFunctionsOfPerson	

Method description: This method retrieves a user's job function and job function groups (JFGs) to which the user belongs. This is applied to all departments. Note that JFGs are not available to members within directory organizations.
 	
C# method call:
`void GetAllJobFunctionsOfPerson(string bstrOrgName, string bstrPerson, out object varJobFunctions)`
 	
VB.NET method call:
`SSub GetAllJobFunctionsOfPerson(ByVal bstrOrgName As String, ByVal bstrPerson As String, ByRef varJobFunctions As Object)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization's name.
`bstrPerson`	**Input Type:** System.String **Description:** `bstrPerson` represents the user's name.
Output	
`varJobFunctions`	**Output Type:** System.Object **Description:** `varJobFunctions` represents the job function names.

GetFullJobFunctionToID

Method: GetFullJobFunctionToID	
Method description: This method retrieves a user's ID based on a specified full job function (in the format department/job function). **C# method call:** void GetFullJobFunctionToID(string bstrOrgName, string bstrFullJobFunction, out string lUserID) **VB.NET method call:** Sub GetFullJobFunctionToID(ByVal bstrOrgName As String, ByVal bstrFullJobFunction As String, ByRef lUserID As String)	
Input	
bstrOrgName	**Input Type:** System.String **Description:** bstrOrgName represents the organization's name.
bstFullJobFunction	**Input Type:** System.String **Description:** bstFullJobFunction represents the full job function name.
Output	
IUserID	**Output Type:** System.String **Description:** IUserID represents the user's ID.

GetFullJobFunctionToName

Method: GetFullJobFunctionToName	
Method description: This method retrieves a user name based on a specified full job function (in the format department/job function). **C# method call:** void GetFullJobFunctionToName(string bstrOrgName, string bstrFullJobFunction, out string bstrUserName2) **VB.NET method call:** Sub GetFullJobFunctionToName(ByVal bstrOrgName As String, ByVal bstrFullJobFunction As String, ByRef bstrUserName2 As String)	
Input	
bstrOrgName	**Input Type:** System.String **Description:** bstrOrgName represents the organization's name.
bstFullJobFunction	**Input Type:** System.String **Description:** bstFullJobFunction represents the full job function name.
Output	
bstrUserName	**Output Type:** System.String **Description:** bstrUserName represents the user's name.

GetPrimaryJobFunction

Method: GetPrimaryJobFunction	

Method description: This method retrieves a user's primary job function based on a specified user name.
 	
C# method call:
`void GetPrimaryJobFunction(string bstrOrgName, string userName, out string JobFunction)`
 	
VB.NET method call:
`Sub GetPrimaryJobFunction(ByVal bstrOrgName As String, ByVal userName As String, ByRef JobFunction As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`userName`	**Input Type:** System.String **Description:** `userName` represents the name of the user whose job function is to be retrieved.
Output	
`JobFunction`	**Output Type:** System.String **Description:** `JobFunction` represents the specified user's job function.

GetUserJobFunctionInDepartment

Method: GetUserJobFunctionInDepartment	

Method description: This method retrieves a specified user's job function within a specified department.
 	
C# method call:
`void GetUserJobFunctionInDepartment(string bstrOrgName, string bstrDepartment, string bstrUserName, out string bstrJobFunction)`
 	
VB.NET method call:
`Sub GetUserJobFunctionInDepartment(ByVal bstrOrgName As String, ByVal bstrDepartment As String, ByVal bstrUserName As String, ByRef bstrJobFunction As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization's name.
`bstrDepartment`	**Input Type:** System.String **Description:** `bstrDepartment` represents the department name.
`bstrUserName`	**Input Type:** System.String **Description:** `bstrUserName` represents the name of the user whose job function in the specified department is to be retrieved.
Output	
`bstrJobFunction`	**Output Type:** System.String **Description:** `bstrJobFunction` represents the user's job function within the specified department.

IDToJobFunction

Method: IDToJobFunction	

Method description: This method retrieves a user's job function based on the user's ID.
 	
C# method call:
`void IDToJobFunction(string bstrOrgName, string bstrUserID, out string bstrJobFunction)`
 	
VB.NET method call:
`Sub IDToJobFunction(ByVal bstrOrgName As String, ByVal bstrUserID As String, ByRef bstrJobFunction As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`bstrUserID`	**Input Type:** System.String **Description:** `bstrUserID` represents the user ID.
Output	
`bstrJobFunction`	**Output Type:** System.String **Description:** `bstrJobFunction` represents the user job function.

JobFunctionToName

Method: JobFunctionToName	

Method description: This method retrieves a person's user name based on a specified job function.
 	
C# method call:
`void JobFunctionToName(string bstrOrgName, string JobFunction, out string shortName)`
 	
VB.NET method call:
`Sub JobFunctionToName(ByVal bstrOrgName As String, ByVal JobFunction As String, ByRef shortName As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`JobFunction`	**Input Type:** System.String **Description:** `JobFunction` represents the job function of the user.
Output	
`shortName`	**Output Type:** System.String **Description:** `shortName` represents the user's short name for the specified job function.

Miscellaneous methods in the `IOrgChart` interface

Outlined below are miscellaneous methods in the `IOrgChart` interface. These methods are listed in alphabetical order:

- *AdvSearch*
- *GetErrorMessage*
- *Init*
- *Search*

AdvSearch

Method: AdvSearch	

Method description: This method executes an advanced search query and retrieves results.
 	
C# method call:
`void AdvSearch(string bstrOrgName, string bstrCustomQuery, object varSearchTypes, object varSearchStrings, object varResultColumns, out object varResults)`
 	
VB.NET method call:
`Sub AdvSearch(ByVal bstrOrgName As String, ByVal bstrCustomQuery As String, ByVal varSearchTypes As Object, ByVal varSearchStrings As Object, ByVal varResultColumns As Object, ByRef varResults As Object)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization's name.
`bstrCustomQuery`	**Input Type:** System.String **Description:** `bstrCustomQuery` represents any custom query.
`varSeachType`	**Input Type:** System.Object **Description:** `varSeachType` represents the search type. The following options are available: • 0: `SEARCH_USER_NAME` • 1: `SEARCH_FULL_NAME` • 2: `SEARCH_JOB_FUNCTION` • 3: `SEARCH_GROUP` • 4: `SEARCH_CHART` • 5: `SEARCH_EMAIL` • 6: `SEARCH_OU` • 7: `SEARCH_USER_ID`
`varSearchString`	**Input Type:** System.Object **Description:** `varSearchString` represents the string to use in the search query.
`varResultColumns`	**Input Type:** System.Object **Description:** `varResultColumns` represents the columns that should appear in the search result.

Method: AdvSearch	
Output	
varResult	**Output Type:** System.Object **Description:** varResult represents a two-dimension string array that represents a table.

GetErrorMessage

Method: GetErrorMessage	
Method description: This method returns error messages. **C# method call:** void GetErrorMessage(out string bstrErrorMsg) **VB.NET method call:** Sub GetErrorMessage(ByRef bstrErrorMsg As String)	
Output	
bstrErrorMsg	**Output Type:** System.String **Description:** bstrErrorMsg represents an error message.

Init

Method: Init	

Method description: This method initializes the components according to organization name.
 	
C# method call:
`void Init(string bstrOrg, string bstrDomain, string bstrRootDSN, string bstrRootOU, string bstrUserName, string bstrPassword, uint ulTimeOut, uint ulPort)`
 	
VB.NET method call:
`Sub Init(ByVal bstrOrg As String, ByVal bstrDomain As String, ByVal bstrRootDSN As String, ByVal bstrRootOU As String, ByVal bstrUserName As String, ByVal bstrPassword As String, ByVal ulTimeOut As UInteger, ByVal ulPort As UInteger)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`bstrDomain`	**Input Type:** System.String **Description:** `bstrDomain` represents the domain name.
`bstrRootDSN`	**Input Type:** System.String **Description:** `bstrRootDSN` represents the root DSN in which to connect.
`bstrRootOU`	**Input Type:** System.String **Description:** `bstrRootOU` represents the root OU.
`bstrUserName`	**Input Type:** System.String **Description:** `bstrUserName` represents the user name for the connection.
`bstrPassword`	**Input Type:** System.String **Description:** `bstrPassword` represents the user password.
`ulTimeOut`	**Input Type:** System.UInt32 **Description:** `ulTimeOut` represents the time out period (in seconds).
`ulPort`	**Input Type:** System.UInt32 **Description:** `ulPort` represents the server port in which to connect.
Return Conditions	
`Init` returns type is void which means it does not return anything.	

Search

Method: Search	

Method description: This method executes a search query and retrieves results.
 	
C# method call:
`void Search(string bstrOrgName, ushort bstrSearchType, string bstrSearchString, out object varResults)`
 	
VB.NET method call:
`Sub Search(ByVal bstrOrgName As String, ByVal bstrSearchType As UShort, ByVal bstrSearchString As String, ByRef varResults As Object)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`bstrSearchType`	**Input Type:** System.UShort **Description:** `bstrSearchType` represents the search type. The following options are available: • 0: `SEARCH_USER_NAME` • 1: `SEARCH_FULL_NAME` • 2: `SEARCH_JOB_FUNCTION` • 3: `SEARCH_GROUP` • 4: `SEARCH_CHART` • 5: `SEARCH_EMAIL` • 6: `SEARCH_OU` • 7: `SEARCH_USER_ID`
`bstrSearchString`	**Input Type:** System.String **Description:** `bstrSearchString` represents the string to use in the search query.
Output	
`varResults`	**Output Type:** System.Object **Description:** `varResults` represents a single-dimension string array of the search results.

"Superior" methods in the IOrgChart interface

Outlined below are the "superior" methods in the IOrgChart interface. These methods are listed in alphabetical order:

- *GetManager*
- *GetManagerFromID*
- *GetManagerJobFunction*
- *GetSupervisor*
- *GetSupervisorFromID*
- *GetSupervisorJobFunction*
- *IsSuperior*

GetManager

Method: GetManager	

Method description: This method retrieves a specified user Manager.
 	
C# method call:
`void GetManager(string bstrOrgName, string userName, out string Manager)`
 	
VB.NET method call:
`Sub GetManager(ByVal bstrOrgName As String, ByVal userName As String, ByRef Manager As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the name of the user whose Manager is to be retrieved.
`userName`	**Input Type:** System.String **Description:** `userName` represents the user's name.
Output	
`Manager`	**Output Type:** System.String **Description:** `Manager` represents the user's Manager.

GetManagerFromID

Method: GetManagerFromID	

Method description: This method retrieves the Manager based on a specified job function ID.
 	
C# method call:
`void GetManagerFromID(string bstrOrgName, string JobFunctionID, out string Manager)`
 	
VB.NET method call:
`Sub GetManagerFromID(ByVal bstrOrgName As String, ByVal JobFunctionID As String, ByRef Manager As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`JobFunctionID`	**Input Type:** System.String **Description:** `JobFunctionID` represents the full job function (in the format `department/job function`).
Output	
`Manager`	**Output Type:** System.String **Description:** `Manager` represents the Manager of the job function ID.

GetManagerJobFunction

Method: GetManagerJobFunction	

Method description: This method retrieves the Manager job function based on a specified Subordinate's job function.
 	
C# method call:
`void GetManagerJobFunction(string bstrOrgName, string JobFunction, out string ManagerJF)`
 	
VB.NET method call:
`Sub GetManager(ByVal bstrOrgName As String, ByVal userName As String, ByRef Manager As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization's name.
`JobFunction`	**Input Type:** System.String **Description:** `JobFunction` represents a Subordinate's job function.
Output	
`ManagerJF`	**Output Type:** System.String **Description:** `ManagerJF` represents the Manager's job function against the specified Subordinate job function.

GetSupervisor

Method: GetSupervisor	

Method description: This method retrieves a specified user's Supervisor.
 	
C# method call:
`void GetSupervisor(string bstrOrgName, string userName, out string Supervisor)`
 	
VB.NET method call:
`Sub GetSupervisor(ByVal bstrOrgName As String, ByVal userName As String, ByRef Supervisor As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`userName`	**Input Type:** System.String **Description:** `userName` represents the name of the user whose Supervisor is to be retrieved.
Output	
`Supervisor`	**Output Type:** System.String **Description:** `Supervisor` represents the specified user's Supervisor.

GetSupervisorFromID

Method: GetSupervisorFromID	

Method description: This method retrieves a Supervisor based on a specified job function ID.
 	
C# method call:
`void GetSupervisorFromID(string bstrOrgName, string JobFunctionID, out string Supervisor)`
 	
VB.NET method call:
`Sub GetSupervisorFromID(ByVal bstrOrgName As String, ByVal JobFunctionID As String, ByRef Supervisor As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization's name.
`JobFunctionID`	**Input Type:** System.String **Description:** `JobFunctionID` represents the full job function (in the format `department/job function`).
Output	
`Supervisor`	**Output Type:** System.String **Description:** `Supervisor` represents the Supervisor of the specified job function ID.

GetSupervisorJobFunction

Method: GetSupervisorJobFunction	

Method description: This method retrieves the Supervisor's job function based on a specified Subordinate's job function.
 	
C# method call:
`void GetSupervisorJobFunction(string bstrOrgName, string JobFunction, out string SupevisorJF)`
 	
VB.NET method call:
`Sub GetSupervisorJobFunction(ByVal bstrOrgName As String, ByVal JobFunction As String, ByRef SupevisorJF As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`JobFunction`	**Input Type:** System.String **Description:** `JobFunction` represents a Subordinate's job function.
Output	
`SupervisorJF`	**Output Type:** System.String **Description:** `SupervisorJF` represents the Supervisor's job function. of the specified user job function.

IsSuperior

Method: IsSuperior	

Method description: This method determines whether a specified person has a Supervisor.
 	
C# method call:
`int IsSuperior(string bstrOrgName, string bstrUserName, string bstrUserSuperior, int bIsCheckPrimary)`
 	
VB.NET method call:
`Function IsSuperior(ByVal bstrOrgName As String, ByVal bstrUserName As String, ByVal bstrUserSuperior As String, ByVal bIsCheckPrimary As Integer) As Integer`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.

Method: IsSuperior	
bstrUserName	**Input Type:** System.String **Description:** bstrUserName represents the user name whose hierarchy is to be searched.
bstrUserSuperior	**Input Type:** System.String **Description:** bstrUserSuperior represents the user name that is to be searched.
bIsCheckPrimary	**Input Type:** System.Int32 **Description:** bIsCheckPrimary represents a primary flag. If **Primary** is TRUE, then only the primary job function's hierarchy is searched.
Return Conditions	
IsSuperior returns 1 if bstrUserSuperior is the Supervisor of bstrUserName. IsSuperior returns 0 if bstrUserSuperior is not the Supervisor of bstrUserName.	

"User information" methods in the IOrgChart interface

Outlined below are the "user information" methods in the IOrgChart interface. These methods are listed in alphabetical order:

- *DoesPersonExist*
- *GetEmailAddress*
- *GetExactCaseUser*
- *GetPersonDepartment*
- *GetUserGroups*
- *GetUserProperties*

DoesPersonExist

Method: DoesPersonExist	

Method description: This method verifies whether a person exists within a specified organization.
C# method call:
`int DoesPersonExist(string bstrOrgName, string userName)`
VB.NET method call:
`Function DoesPersonExist(ByVal bstrOrgName As String, ByVal userName As String) As Integer`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`userName`	**Input Type:** System.String **Description:** `userName` represents the name of the user being sought.
Return Conditions	
`DoesPersonExist` returns `1` if the person does exist in the organization. `DoesPersonExist` returns `0` if the person does not exist in the organization.	

GetEmailAddress

Method: GetEmailAddress	

Method description: This method retrieves a specified user's e-mail address.
C# method call:
`void GetEmailAddress(string userName, out string eMail)`
VB.NET method call:
`Sub GetEmailAddress(ByVal userName As String, ByRef eMail As String)`	

Input	
`userName`	**Input Type:** System.String **Description:** `userName` represents the name of the user.
Output	
`eMail`	**Output Type:** System.String **Description:** `eMail` represents the user's e-mail address.

GetExactCaseUser

Method: GetExactCaseUser	

Method description: This method searches for the name of a user or a group.
 	
C# method call:
`void GetExactCaseUser(string bstrOrgName, string bstrUserName, out string pbstrExactUser)`
 	
VB.NET method call:
`Sub GetExactCaseUser(ByVal bstrOrgName As String, ByVal bstrUserName As String, ByRef pbstrExactUser As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`bstrUserName`	**Input Type:** System.String **Description:** `bstrUserName` represents the name of the user whose exact case is to be determined.
Output	
`pbstrExactUser`	**Output Type:** System.String **Description:** `pbstrExactUser` represents the user's name if a valid name is specified. Otherwise, the method searches for a valid group and `pbstrExactUser` returns a group name. If both fail, the User ID is returned.

GetPersonDepartment

Method: GetPersonDepartment	

Method description: This method retrieves the name of the first department a specified user is found within a specified organization.
 	
C# method call:
`void GetPersonDepartment(string bstrOrgName, string bstrUserID, out string dept)`
 	
VB.NET method call:
`Sub GetPersonDepartment(ByVal bstrOrgName As String, ByVal bstrUserID As String, ByRef dept As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`bstrUserID`	**Input Type:** System.String **Description:** `bstrUserID` represents the person's user name.
Output	
`dept`	**Output Type:** System.String **Description:** `dept` represents the department in which the user is a member.

GetUserGroups

Method: GetUserGroups	

Method description: This method retrieves the groups and departments in which a specified user is a member. This method also retrieves the job functions in which that user serves within those departments.
 	
C# method call:
`void GetUserGroups(string bstrOrgName, string bstrUserName, out object varGroups, out object varDepts, out object varJFGs)`
 	
VB.NET method call:
`Sub GetUserGroups(ByVal bstrOrgName As String, ByVal bstrUserName As String, ByRef varGroups As Object, ByRef varDepts As Object, ByRef varJFGs As Object)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`bstrUserName`	**Input Type:** System.String **Description:** `bstrUserName` represents the name of the user.
Output	
`varGroups`	**Output Type:** System.Object **Description:** `varGroups` represents an array of group names in an object.
`varDepts`	**Output Type:** System.Object **Description:** `varDepts` represents an array of department names in an object.
`varJFGs`	**Output Type:** System.Object **Description:** `varJFGs` represents an array of job function groups in an object.

GetUserProperties

Method: GetUserProperties	

Method description: This method retrieves a specified user's properties.
 	
C# method call:
`void GetUserProperties(string bstrOrgName, string bstrUserName, out string bstrFullName, out string bstrJobFunction, out string bstrUserID, out string bstrSupervisor, out string bstrManager, out string bstrDepartment)`
 	
VB.NET method call:
`Sub GetUserProperties(ByVal bstrOrgName As String, ByVal bstrUserName As String, ByRef bstrFullName As String, ByRef bstrJobFunction As String, ByRef bstrUserID As String, ByRef bstrSupervisor As String, ByRef bstrManager As String, ByRef bstrDepartment As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`bstrUserName`	**Input Type:** System.String **Description:** `bstrUserName` represents the name of the user.
Output	
`bstrFullNames`	**Output Type:** System.String **Description:** `bstrFullNames` represents the specified user's full name.
`bstrJobFunction`	**Output Type:** System.String **Description:** `bstrJobFunction` represents the specified user's job function.
`bstrUserID`	**Output Type:** System.String **Description:** `bstrUserID` represents the specified user's ID.
`bstrSupervisor`	**Output Type:** System.String **Description:** `bstrSupervisor` represents the specified user's Supervisor.
`bstrManager`	**Output Type:** System.String **Description:** `bstrManager` represents the specified user's Manager.
`bstrDepartment`	**Output Type:** System.String **Description:** `bstrDepartment` represents the name of the department to which the specified user belongs.

"User name" methods in the IOrgChart interface

Outlined below are the "user name" methods in the IOrgChart interface. These methods are listed in alphabetical order:

- *FullNameToShortName*
- *IdToShortName*
- *ShortNameToFullName*
- *ShortNameToId*

FullNameToShortName

Method: FullNameToShortName	

Method description: This method retrieves a user's short name based on a specified full name.
C# method call:
`void FullNameToShortName(string bstrOrgName, string FullName, out string shortName)`
VB.NET method call:
`Sub FullNameToShortName(ByVal bstrOrgName As String, ByVal FullName As String, ByRef shortName As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`FullName`	**Input Type:** System.String **Description:** `FullName` represents the user's full name.
Output	
`shortName`	**Output Type:** System.String **Description:** `shortName` represents the user short name of the specified user.

IdToShortName

Method: IdToShortName	

Method description: This method retrieves a user's short name based on a specified user ID.
 	
C# method call:
`void IdToShortName(string bstrOrgName, string bstrUserID, out string userName)`
 	
VB.NET method call:
`Sub IdToShortName(ByVal bstrOrgName As String, ByVal bstrUserID As String, ByRef userName As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`bstrUserID`	**Input Type:** System.String **Description:** `bstrUserID` represents the user's ID.
Output	
`userName`	**Output Type:** System.String **Description:** `userName` represents the user short name of the specified user ID.

ShortNameToFullName

Method: ShortNameToFullName	

Method description: This method retrieves a user's full name based on a specified user name.
 	
C# method call:
`void ShortNameToFullName(string bstrOrgName, string shortName, out string FullName)`
 	
VB.NET method call:
`Sub ShortNameToFullName(ByVal bstrOrgName As String, ByVal shortName As String, ByRef FullName As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`shortName`	**Input Type:** System.String **Description:** `shortName` represents the user's short name (in the format `domain/name`).
Output	
`FullName`	**Output Type:** System.String **Description:** `FullName` represents the user's full name.

ShortNameToId

Method: ShortNameToId	

Method description: This method retrieves a user's ID based on a specified user name.
C# method call:
`void ShortNameToId(string bstrOrgName, string person, out string id)`
VB.NET method call:
`Sub ShortNameToId(ByVal bstrOrgName As String, ByVal person As String, ByRef id As String)`	

Input	
`bstrOrgName`	**Input Type:** System.String **Description:** `bstrOrgName` represents the organization name.
`person`	**Input Type:** System.String **Description:** `person` represents the name of the person whose user ID is required.
Output	
`Id`	**Output Type:** System.String **Description:** `Id` represents the user's ID.

Installing the Ultimus Custom OC interface

This section outlines how to install the Ultimus Custom OC interface.

These instructions assume the following prerequisites:

- A custom organizational chart (OC) database has been configured.
- A `SharedInterfaceLib.dll` interface file has been programmed and compiled.
- You are ready to install the Custom OC interface.

In the following discussion, the following instructions assume your custom OC interface `DLL` file is named `OCSyncInterface.dll`. The `SharedInterfaceLib` is the interface file which contains the methods used within the Custom OC interface.

Note: To help demonstrate how a third-party custom database can be utilized in Ultimus Adaptive BPM Suite, the Ultimus EIK includes a sample OC database and a custom OC interface `DLL` file which can be installed. This may be downloaded from the Ultimus Web site. Download the Ultimus EIK sample called "Using a custom organization chart (OC)."

In overview, these are the procedures to install the Ultimus Custom OC interface:

1. *Creating a DSN pointing to the custom OC database*
2. *Registering* `OCSyncInterface.dll` *on Ultimus BPM Server for a VB.NET or C# solution*
3. *Configuring Ultimus System Administrator to work with the new database*

These three broad steps are graphically represented in Figure 24.

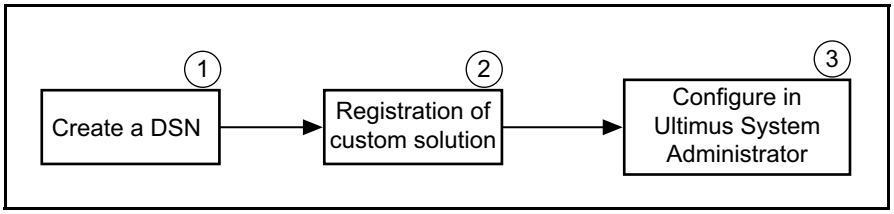

Figure 24. Outlining the Ultimus Custom OC interface installation procedure

Each of these procedures is discussed below.

Creating a DSN pointing to the custom OC database

Create a DSN to your custom OC database if your custom solution is utilizing an ODBC interface to interact with your custom organizational database. Call this DSN `OC Sync`.

Tip: If your custom solution has embedded an OLEDB interface call, then creating the DSN, as specified in this section, is not necessary.

Registering `OCSyncInterface.dll` on Ultimus BPM Server for a VB.NET or C# solution

The next step, when using a VB.NET or C# solution, is to register the custom `OCSyncInterface.dll` file on the computer hosting Ultimus BPM Server.

To register the custom `OCSyncInterface.dll` file, follow these steps:

1. From the computer hosting Ultimus BPM Server, open a command prompt window. To do so, go to **Start»Run**, then enter `cmd`.
2. Browse to *`hard drive:\Windows directory`*`\Microsoft.NET\Framework\v2.0.50727` where *`hard drive:\Windows directory`* represents the hard drive on which the Windows operating system is installed. An example is shown in Figure 25.

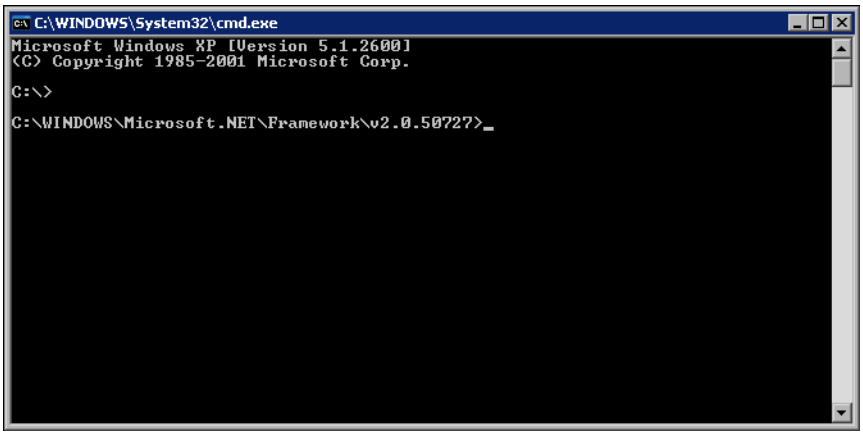

Figure 25. Using the command prompt window

3. Using the registration command, enter:

   `regasm "`*`path of OCSyncInterface.dll`*`" /codebase`

   `path of OCSyncInterface.dll` represents the directory path where `OCSyncInterface.dll` is stored on the computer hosting Ultimus BPM Server. Note that this is case-sensitive. An example is offered below:

   ```
   regasm "C:\Program Files\Ultimus Adaptive BPM Suite 8.3
   \OCSyncInterface.dll" /codebase
   ```

4. The following message displays after successfully registering `OCSyncInterface.dll`, as shown in Figure 26.

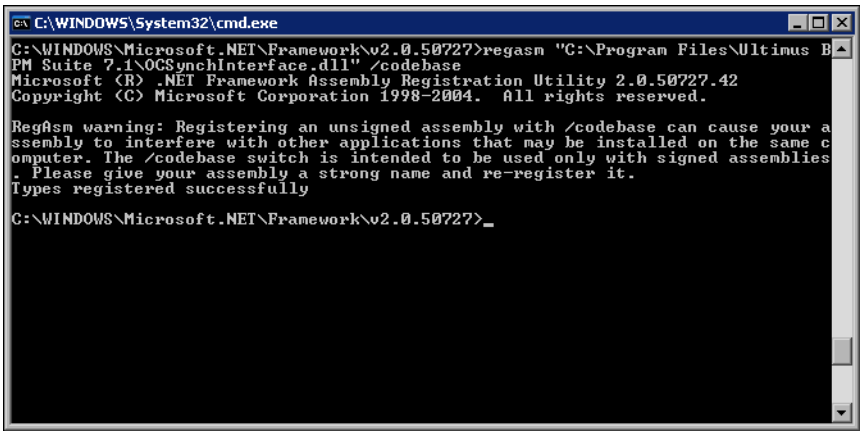

Figure 26. Successfully registering `OCSyncInterface.dll`

5. After successfully registering `OCSyncInterface.dll`, the command prompt window may be closed. This completes the registration of `OCSyncInterface.dll`.

Configuring Ultimus System Administrator to work with the new database

Once the custom solution has been registered on the computer hosting Ultimus BPM Server, Ultimus System Administrator must be configured to interact with the newly created COM+ application.

To configure Ultimus System Administrator to interact with a COM+ application, follow these steps:

1. Open Ultimus System Administrator. To do so, go to **Start»Programs»Ultimus Adaptive BPM Suite 8.3»Ultimus System Administrator**.
2. Expand the Ultimus BPM Server node.
3. Right-click on the **Organizations** node, then select **Add Directory Organization**. The **Directory Configuration** dialog box appears.

4. Select the **Custom** option, as shown in Figure 27.

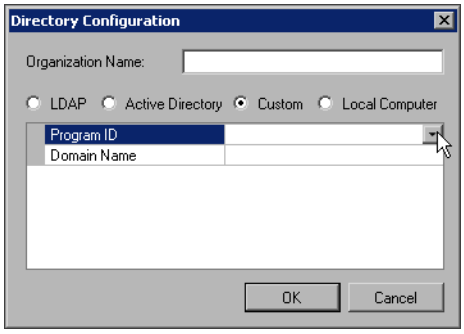

Figure 27. Directory Configuration dialog box

5. From the **Program ID** combo box, select the appropriate COM+ component; if you have implemented the Ultimus EIK Custom OC interface sample, the name would appear as `OCSyncInterface.VBOSync`.
6. Click the **OK** button.
7. It is advised to either reboot the computer hosting Ultimus BPM Server or shut down the Ultimus COM+ package in **Component Services**. Either one of these steps is necessary to reinitialize the Ultimus BPM environment.

Ultimus Adaptive BPM Suite now interfaces with the selected database and all user information is retrieved from this database. Information stored in the external database is then reflected in all Ultimus Adaptive BPM Suite modules wherever Ultimus Organization Charts information would have been displayed.

9

Customizing Ultimus BPM Server e-mail messages

This section discusses how to customize Ultimus BPM Server e-mail messages. By default, Ultimus BPM Server integrates with Microsoft Exchange Server in order to process and deliver e-mail notifications and messages to Ultimus Adaptive BPM Suite users. To extend Ultimus e-mail functionality, users can integrate the Ultimus Adaptive BPM Suite using any third-party e-mail system.

To allow the Ultimus Adaptive BPM Suite to work with any third-party e-mail system, a standard Windows `DLL` file is required. There are four characteristics of the custom `DLL` file which must be followed:

- The custom e-mail notification or auto-launch `DLL` file must be developed in C++. It cannot be developed in C#.
- All method calls described in the following section must be incorporated into any custom e-mail notification or auto-launch `DLL` file.
- Ultimus Adaptive BPM Suite passes Unicode strings to the custom `DLL` file. Ensure to compile and build your `DLL` file using the Unicode character set.
- When creating a custom `DLL` file for e-mail notification, use 30 characters or fewer when setting parameter values. Ultimus Adaptive BPM Suite limits parameter values to 30 characters.

Once developed, the Advanced Server Configuration application can be used to target the custom e-mail notification `DLL` file for use by Ultimus BPM Server.

The following method calls must be provided for any custom e-mail notification `DLL` to be used by Ultimus BPM Server (outlined in alphabetical order):

- *Free*
- *GetInitParams*
- *GetLastErr*
- *GetLastErrHr*
- *Init*
- *Login*
- *LogOff*
- *SendMail*
- *SendMails*
- *ReadLaunchMails*

Free

Method: Free	
Method description: This method frees the object pointer in its parameter. **C++ method call:** `void Free(hObj)`	
Input	
`hObj`	**Input Type:** DWORD **Description:** `hObj` points to the mail object returned in *Init* method (outlined on page 391).
Return Conditions	
`Free` returns `TRUE` if `Free` was able to be executed. `Free` returns `FALSE` if `Free` could not be executed.	

GetInitParams

Method: GetInitParams	
Method description: This method is called by Ultimus to acquire a list of parameters the `DLL` file needs to initialize itself. **C++ method call:** `void GetInitParams(**parray)`	
Output	
`parray`	**Output Type:** VT_ARRAY|VT_BSTR **Description:** `parray` contains the initialization parameters that the custom e-mail `DLL` file needs to initialize the e-mail system (such as the mail server name, log in name, and password).
Return	
Description: The `DLL` file should return these variables in a VARIANT containing an array of BSTRs. These are presented to the user in Advanced Configuration Application when setting up custom e-mail for notification or for e-mail auto-launch.	

GetLastErr

Method: GetLastErr	
Method description: This method retrieves the last error on the e-mail server, then returns the error message as a string. **C++ method call:** `void GetLastErr(hObj, IpstrErr, nSize)`	
Input	
`hObj`	**Input Type:** DWORD **Description:** `hObj` points to the mail object returned in *Init* method (outlined on page 391).
`IpstrErr`	**Input Type:** LPCTSTR **Description:** `IpstrErr` points to a string array that receives the error from the e-mail server.
`nSize`	**Input Type:** UINT **Description:** `nSize` specifies the size of the buffer for `IPstrErr` parameter.

GetLastErrHr

Method: GetLastErrHr	
Method description: This method retrieves the last error on the e-mail server, then returns the error message in `hresult`. **C++ method call:** `hresult GetLastErrHr (hObj)`	
Input	
`hObj`	**Input Type:** DWORD **Description:** `hObj` points to the mail object returned in *Init* method (outlined on page 391).
Return Conditions	
Output Type: HResult **Description:** A return of `TRUE` occurs if the method successfully retrieves the last error on the e-mail server, then returns the error message in hresult; otherwise, it returns `FALSE`.	

Init

Method: Init	
Method description: This method initializes the DLL file for either e-mail notification or auto-launch. **C++ method call:** `dword Init(bUI, Ipcookie)`	
Input	
bUI	**Input Type:** bool **Description:** bUI has different meanings in different scenarios. bUI can have the following meanings: • If the e-mail server is MAPI (such as Microsoft Exchange), the e-mail server is to initialize as a single-threaded or free-threaded object model. If the custom e-mail DLL file is a MAPI DLL, this parameter should be used to initialize the MAPI e-mail server. • If the e-mail server is SMTP or POP, this parameter is set to zero, and this parameter can be ignored.
Ipcookie	**Input Type:** LPCTSTR **Description:** Ipcookie can be set to one of two values that tells the DLL file to initialize itself either for e-mail notification (NOTIFICATION) or auto-launch (AUTOLAUNCH). The DLL file should read the appropriate parameters from the registry to initialize itself. • The NOTIFICATION settings are stored in HKEY_LOCAL_MACHINE\SOFTWARE\Ultimus\Server\Notification*EMAILDLLNAME*. • The AUTOLAUNCH settings are stored in HKEY_LOCAL_MACHINE\SOFTWARE\Ultimus Server\AutoLaunchSettings*EMAILDLLNAME*.
Return	
Output Type: DWORD **Description:** The return parameter is a mail object pointer that is to be used in subsequent calls to send or receive e-mail.	

Login

Method: Login	
Method description: This method is used to log on the e-mail server. **C++ method call:** `bool Login(hObj, strUser, strPassword)`	
Input	
`hObj`	**Input Type:** DWORD **Description:** `hObj` points to the mail object returned in *Init* method (outlined on page 391).
`strUser`	**Input Type:** LPCTSTR **Description:** `strUser` represents the log in name (or ID) of the user.
`strPassword`	**Input Type:** LPCTSTR **Description:** `strPassword` represents the user's password.
Return Conditions	
`Login` returns `TRUE` if `Login` was able to be executed. `Login` returns `FALSE` if `Login` could not be executed or if the user could not be verified.	

LogOff

Method: LogOff	
Method description: This method logs off from the e-mail server. **C++ method call:** `bool LogOff(hObj)`	
Input	
`hObj`	**Input Type:** DWORD **Description:** `hObj` points to the mail object returned in *Init* method (outlined on page 391).
Return Conditions	
`Logoff` returns `TRUE` if `Logoff` was able to be executed. `Logoff` returns `FALSE` if `Logoff` could not be executed.	

SendMail

Method: SendMail	
Method description: This method sends e-mail to a single recipient. **C++ method call:** `bool SendMail(DWORD hObj, LPCTSTR strTo, LPCTSTR szCC, LPCTSTR szBCC, LPCTSTR strSubject, LPCTSTR strBody, SAFEARRAY *parray, bool bHtmlBody)`	
Input	
`hObj`	**Input Type:** DWORD **Description:** `hObj` points to the mail object returned in *Init* method (outlined on page 391).
`strTo`	**Input Type:** LPCTSTR **Description:** `strTo` represents the e-mail address of the recipient.
`strSubject`	**Input Type:** LPCTSTR **Description:** `strSubject` represents the subject (line) of the e-mail.
`strBody`	**Input Type:** LPCTSTR **Description:** `strBody` represents the body (or content) of the e-mail.

Method: SendMail	
Output	
`parray`	**Output Type:** VT_BSTR **Description:** `parray` contains a list of attachments (direct path directories to the file attachments).
Return Conditions	
`SendMail` returns `TRUE` if `SendMail` was able to be executed. `SendMail` returns `FALSE` if `SendMail` could not be executed.	

SendMails

Method: SendMails	
Method description: This method sends e-mail to multiple recipients. **C++ method call:** `bool SendMails(DWORD hObj, SAFEARRAY *pToarray, SAFEARRAY *pCCarray, SAFEARRAY *pBCCarray, LPCTSTR strSubject, LPCTSTR strBody, SAFEARRAY *parray, bool bHtmlBody)`	
Input	
`hObj`	**Input Type:** DWORD **Description:** `hObj` points to the mail object returned in *Init* method (outlined on page 391).
`strSubject`	**Input Type:** LPCTSTR **Description:** `strSubject` represents the subject (line) of the e-mail.
`strBody`	**Input Type:** LPCTSTR **Description:** `strBody` represents the body (or content) of the e-mail.
Output	
`pToarray`	**Output Type:** VT_ARRAY|VT_BSTR **Description:** `*pToarray` is a variant containing an array of BSTRS of e-mail addresses of the e-mail's multiple recipients.
`pArray`	**Output Type:** VT_BSTR **Description:** `pArray` contains a list of attachments (direct path directories to the file attachments).

Method: SendMails
Return Conditions
`SendMails` returns `TRUE` if `SendMails` was able to be executed.
`SendMails` returns `FALSE` if `SendMails` could not be executed.

ReadLaunchMails

Method: ReadLaunchMails	
Method description: This method reads the auto-launch e-mails from the user's inbox. For performance reasons, this method should not process more than 50 e-mails at a time. **C++ method call:** `bool ReadLaunchMails(hObj)`	
Input	
`hObj`	**Input Type:** DWORD **Description:** `hObj` points to the mail object returned in *Init* method (outlined on page 391).
Return Conditions	
`ReadLaunchMails` returns `TRUE` if `ReadLaunchMails` was able to be executed.	
`ReadLaunchMails` returns `FALSE` if `ReadLaunchMails` could not be executed.	

10

The Ultimus Form Control Object

This section discusses the Ultimus Form Control Object (FCO), its methods, and how to insert a third-party control. The Ultimus FCO is a .Net User control that is hosted in Internet Explorer. The FCO controls the data transfer between the controls in a step's Ultimus Form and its corresponding step XML schema within a business process.

The Ultimus FCO is discussed in the following sections:

- *Ultimus Form Control Object methods and events*
- *Methods to manipulate recordsets*
- *Methods pertaining to form actions*
- *Methods to manipulate XML schema*
- *Methods to manipulate controls*
- *Extending the functionality of the Ultimus Database Grid and Schema Grid controls*
- *Inserting a Place Holder control*
- *Example of a custom third-party control*
- *Providing an ActiveX wrapper to a .Net user control*
- *Distributing and using third-party controls*

Ultimus Form Control Object methods and events

Outlined below are all the methods and events within the Ultimus Form Control Object (FCO). For each method there is a description of the method, its method call (written in JavaScript), its parameters, comments (where applicable), and an example. These methods are as follows (in alphabetical order):

- *ClearVariable*
- *ControlValueChanged*
- *FormReturned*
- *FormSent*
- *GetElementValue*
- *GetIncidentNo*

- *GetProcessName*
- *GetStepLabel*
- *GetTaskID*
- *GetTaskStatus*
- *GetTaskSubStatus*
- *GetTaskUser*
- *GetUserFullName*
- *GetVarValue*
- *GetVarValuesArray*
- *IsInRealTime*
- *IsInSimulation*
- *IsInTest*
- *PageSwitched*
- *PreFormReturn*
- *PreFormSend*
- *SetElementValue*
- *SetVarValue*
- *SetVarValueRef*
- *SetVarValuesArray*

Caution: Only the methods outlined below are supported for customer and partner use; Ultimus does not support the use of any FCO method that is not documented in this section.

ClearVariable

Method: ClearVariable	
Method description: ClearVariable sets the element value to null. **JavaScript method call:** `bool ClearVariable(string VarName)`	
Input	
`VarName`	**Input Type:** String **Description:** `VarName` is name of the element to be cleared.

ControlValueChanged

Method: ControlValueChanged	
Method description: `ControlValueChanged` notifies the FCO that control value has changed so that it can perform the necessary operation (such as set the changed value in a control's destination). **JavaScript method call:** `void ControlValueChanged(object ObjCtrl)`	
Input	
`ObjCtrl`	**Input Type:** Object **Description:** `ObjCtrl` is the HTML DOM element object.

FormReturned

Method: FormReturned
Method description: `FormReturned` event is triggered after the form is returned to Ultimus BPM Server. **JavaScript method call:** `FormReturned()`

FormSent

Method: FormSent
Method description: `FormSent` event is triggered after the form is sent to Ultimus BPM Server. **JavaScript method call:** `FormSent()`

GetElementValue

Method: GetElementValue	
Method description: `GetElementValue` retrieves the value of a specific schema element. **JavaScript method call:** `public object GetElementValue(string strXPath)`	
Input	
`strXPath`	**Input Type:** String **Description:** `strXPath` represents the XPath of the element to be retrieved.

Example

```
// This code snippet is written in JavaScript
window.parent.frames(0).document.theFCO.GetElementValue('TaskData.variable');
// Where TaskData.variable is the XPath of the element.
```

Comments

If the element is of simple type then the method returns the object of XML schema element type containing the value of the element.

If the element is of simple repeated type then the method returns the JavaScript array object of XML schema element type containing the values of the element.

If element is of complex repeated type then the method returns the XML as string containing all rows of element.

This method can also take index `xpath`. This index path can be used in both simple and complex repeated elements. Elements cannot be inserted through the index because this is only to retrieve a value if an XML schema element already exists. These indexes are zero based.

For example, if `TaskData.variable` is a simple repeated element, then `xpath TaskData.variable[0]` returns a single value at 0 index.

GetIncidentNo

Method: GetIncidentNo
Method description: `GetIncidentNo` returns the incident number of the process to which the task's form belongs. **JavaScript method call:** `int GetIncidentNo()`
Example
`// This code snippet is written in JavaScript` `window.parent.frames(0).document.theFCO.GetIncidentNo();`

GetProcessName

Method: GetProcessName
Method description: `GetProcessName` returns the task process name to which the form belongs. This method returns the correct value when the form is viewed in simulation or real-time modes. **JavaScript method call:** `string GetProcessName()`
Example
`// This code snippet is written in JavaScript` `window.parent.frames(0).document.theFCO.GetProcessName();`

GetStepLabel

Method: GetStepLabel
Method description: `GetStepLabel` returns the task step label to which the form belongs. **JavaScript method call:** `string GetStepLabel()`
Example
`// This code snippet is written in JavaScript` `window.parent.frames(0).document.theFCO.GetStepLabel();`

GetTaskID

Method: GetTaskID

Method description: `GetTaskID` returns the task id to which the form belongs.

JavaScript method call:

```
string GetTaskID()
```

Example

```
// This code snippet is written in JavaScript
window.parent.frames(0).document.theFCO.GetTaskID();
```

GetTaskStatus

Method: GetTaskStatus

Method description: `GetTaskStatus` returns the main status of the task to which the form belongs. The possible output values are listed below:

Output value	Task Status
1	Active task
3	Complete task
4	Returned task
7	Aborted task

JavaScript method call:

```
int GetTaskStatus()
```

Example

```
// This code snippet is written in JavaScript
window.parent.frames(0).document.theFCO.GetTaskStatus();
```

GetTaskSubStatus

Method: GetTaskSubStatus

Method description: `GetTaskSubStatus` returns the sub-status of the task to which the form belongs. The possible output values are listed below:

Output value	Task Status
`TASK_STATUS_ACTIVE_LATE`	0x1
`TASK_STATUS_ACTIVE_OVERDUE`	0x2
`TASK_STATUS_ACTIVE_REASSIGNED`	0x4
`TASK_STATUS_ACTIVE_CONFERED`	0x8
`TASK_STATUS_ACTIVE_RETURNED`	0x10

JavaScript method call:

```
int GetTaskSubStatus()
```

Example

```
// This code snippet is written in JavaScript
window.parent.frames(0).document.theFCO.GetTaskSubStatus();
```

GetTaskUser

Method: GetTaskUser

Method description: `GetTaskUser` returns the user name of the individual who has opened the task. The format of the returned user name is a fully resolved user id (in the format `mycompany.com/jdoe`).

JavaScript method call:

```
string GetTaskUser()
```

Example

```
// This code snippet is written in JavaScript
window.parent.frames(0).document.theFCO.GetTaskUser();
```

GetVarValue

Method: GetVarValue	
Method description: `getVarValue` returns the value of a specified schema element. **JavaScript method call:** `string getVarValue(varName)`	
Input	
`varName`	**Input Type:** String **Description:** `varName` is the name of an XML schema element.
Example	
`// This code snippet is written in JavaScript` `window.parent.frames(0).document.theFCO.GetVarValue(Taskdata.Global.MySchemaElement);`	
Comments	
This method is for backward compatibility with Ultimus BPM Suite 7.*x* and continues to be supported as long as it is used to get the value of a simple node. The method is not type based, which means it returns "string" value no matter what is the type of element. It is strongly recommended to use *SetElementValue* (on page 407) instead of this method.	

GetUserFullName

Method: GetUserFullName
Method description: `GetUserFullName` returns the task's user full name. **JavaScript method call:** `string GetUserFullName()`

GetVarValuesArray

Method: GetVarValuesArray	
Method description: `GetVarValuesArray` returns an Object of Type Array having the values of the specified XML schema element. **JavaScript method call:** `public object GetVarValuesArray(string VarName)`	
Input	
`strCPH`	**Input Type:** String **Description:** `strCPH` represents `th`e control name. This string is optional; it may be sent as an empty string.
`varName`	**Input Type:** String **Description:** `varName` is a simple repeated element name.
Example	
`// This code snippet is written in JavaScript` `window.parent.frames(0).document.theFCO.GetVarValuesArray("TaskData.Global.ListItems")`	
Comments	
The returned values are type-based Array; that is, the type of the returned array depends upon the type of the specified element.	

IsInRealTime

Method: IsInRealTime
Method description: `IsInRealTime` returns `TRUE` if the form is opened in real time mode. Otherwise, `IsInRealTime` returns `FALSE`. **JavaScript method call:** `bool IsInRealTime()`
Example
`// This code snippet is written in JavaScript` `window.parent.frames(0).document.theFCO.IsInRealTime();`

IsInSimulation

Method: IsInSimulation
Method description: `IsInSimulation` returns `TRUE` if the form is opened in simulation mode. Otherwise, `IsInSimulation` returns `FALSE`. **JavaScript method call:** `bool IsInSimulation()`
Example
`// This code snippet is written in JavaScript` `window.parent.frames(0).document.theFCO.IsInSimulation();`

IsInTest

Method: IsInTest
Method description: `IsInTest` returns `TRUE` if the form is opened in test mode. Otherwise, `IsInTest` returns `FALSE`. **JavaScript method call:** `bool IsInTest()`
Example
`// This code snippet is written in JavaScript` `window.parent.frames(0).document.theFCO.IsInTest();`

PageSwitched

Method: PageSwitched	
Method description: `PageSwitched` event is triggered when the user selects a new page in the Ultimus Form. **JavaScript method call:** `PageSwitched(pageName, filename)`	
Input	
`pageName`	**Input Type:** String **Description:** `pageName` is the name of the page the user has selected.
`fileName`	**Input Type:** String **Description:** `fileName` is the name of the HTML file that should be loaded.

PreFormReturn

Method: PreFormReturn
Method description: `PreFormReturn` event is triggered when the "Return" button is clicked by the user before the form is returned to Ultimus BPM Server. **JavaScript method call:** `PreFormReturn()`

PreFormSend

Method: PreFormSend
Method description: `PreFormSend` event is triggered when the "Send" button is clicked by the user before the form is returned to Ultimus BPM Server. **JavaScript method call:** `PreFormSend()`

SetElementValue

Method: PreFormSend	
Method description: `SetElementValue` sets the value of a specific schema element. This method returns `TRUE` if the value is set successfully; otherwise this method returns `FALSE`. **JavaScript method call:** `public bool SetElementValue(string strXPath, object ObjValue)`	
Input	
`strXPath`	**Input Type:** String **Description:** `strXPath` represents the XPath of the element.
Example	

`// This code snippet is written in JavaScript`
`window.parent.frames(0).document.theFCO.SetElementValue('TaskData.variable', 'strValue');`
`// Where TaskData.variable is the element XPath.`	

Comments	
`SetElementValue` can set the values for all the element types: • For simple types the object passed as the second parameter should be of the type of the simple element. • For a simple repeated element, the second parameter should be an array of the type of the repeated element. • For a complex type or complex repeated type, the object passed as the second parameter should be XML representing those types. This method can also take index `xpath`. This index path can be used in both simple and complex repeated elements. Elements cannot be inserted through the index because this is only to set a value if an XML schema element already exists. These indexes are zero based. For example, if `TaskData.variable` is a simple repeated element, then `xpath TaskData.variable[0]` returns a single value at 0 index.	

SetVarValue

Method: SetVarValue	
Method description: `setVarValue` sets the value of the specified XML schema element. **JavaScript call:** `bool setVarValue(varName, varValue)`	
Input	
`varName`	**Input Type:** String **Description:** `varName` is the name of the XML schema element.
`varValue`	**Input Type:** String **Description:** `varValue` is to be written.
Example	
`// This code snippet is written in JavaScript` `window.parent.frames(0).document.theFCO.SetVarValue("varName", document.forms[0].EditBox1.value);`	
Comments	
This method is for backward compatibility with Ultimus BPM Suite 7.*x* and continues to be supported as long as it is used to get the value of a simple node. The method is not type based, which means it returns "string" value no matter what is the type of element. It is strongly recommended to use *SetElementValue* (on page 407) instead of this method.	

SetVarValueRef

Method: SetVarValueRef	
Method description: `SetVarValueRef` acquires the value from the source element and sets it in the destination element provided the source and destination elements are of the same type. **JavaScript call:** `bool SetVarValueRef(string DestCells, string SrcCells)`	
Input	
`DestCells`	**Input Type:** String **Description:** `DestCells` is the destination element.
`SrcCells`	**Input Type:** String **Description:** `DestCells` is the source element.

SetVarValuesArray

Method: SetVarValuesArray	
Method description: `SetVarValuesArray` sets the value of a specified repeated XML schema element. **JavaScript call:** `public bool SetVarValuesArray(string VarName, object ObjValues)`	
Input	
`varName`	**Input Type:** String **Description:** `varName` is a simple repeated element name.
`ObjValues`	**Input Type:** Object **Description:** `ObjValues` is an Object of Type Array having the values to set against the specified schema element. It should be the type-based Array, depending upon the type of schema element against which the value is to be set.
Example	

```
// This code snippet is written in JavaScript
window.parent.frames(0).document.theFCO.SetVarValuesArray("TaskData.Global.
ListItems",values);
```

Methods to manipulate recordsets

This section outlines methods used to manipulate recordsets. For each method there is a description of the method, its method call (written in JavaScript), its parameters, and comments (where applicable). These methods are as follows (in alphabetical order):

- *RSAddNew*
- *RSClear*
- *RSDelete*
- *RSExecDBA*
- *RSMoveFirst*
- *RSMoveLast*
- *RSMoveNext*
- *RSMovePrev*
- *RSMoveTo*
- *RSRefresh*
- *RSUpdate*

RSAddNew

Method: RSAddNew	
Method description: `RSAddNew` adds a new row in the recordset. The newly added row becomes the current row for the given recordset. **JavaScript call:** `void RSAddNew(string RSName)`	
Input	
`RSName`	**Input Type:** String **Description:** `RSName` is the recordset name.
Comment	
The method is not synchronized and executes in a separate thread.	

RSClear

Method: RSClear	
Method description: `RSClear` clears the control values linked with the recordset. **JavaScript call:** `void RSClear(string RSName)`	
Input	
`RSName`	**Input Type:** String **Description:** `RSName` is the recordset name.
Comment	
The method is not synchronized and executes in a separate thread.	

RSDelete

Method: RSDelete	
Method description: `RSDelete` deletes the current row in the recordset. The previous row in the recordset becomes the current row. **JavaScript call:** `void RSDelete(string RSName)`	
Input	
`RSName`	**Input Type:** String **Description:** `RSName` is the recordset name.
Comment	
The method is not synchronized and executes in a separate thread.	

RSExecDBA

Method: RSExecDBA	
Method description: `RSExecDBA` executes a specified SQL action. **JavaScript call:** `bool RSExecDBA(string Connection, string DBAction)`	
Input	
`Connection`	**Input Type:** String **Description:** `Connection` is the connection name.
`DBAction`	**Input Type:** String **Description:** `DBAction` is the SQL action name.
Comment	
The method is not synchronized and executes in a separate thread.	

RSMoveFirst

Method: RSMoveFirst	
Method description: `RSMoveFirst` moves the recordset cursor to the first position. **JavaScript call:** `void RSMoveFirst(string RSName)`	
Input	
`RSName`	**Input Type:** String **Description:** `RSName` is the recordset name.
Comment	
The method is not synchronized and executes in a separate thread.	

RSMoveLast

Method: RSMoveLast	
Method description: `RSMoveLast` moves the recordset cursor to the last position. **JavaScript call:** `void RSMoveLast(string RSName)`	
Input	
`RSName`	**Input Type:** String **Description:** `RSName` is the recordset name.
Comment	
The method is not synchronized and executes in a separate thread.	

RSMoveNext

Method: RSMoveNext	
Method description: `RSMoveNext` moves the recordset cursor to the next position. **JavaScript call:** `void RSMoveNext(string RSName)`	
Input	
`RSName`	**Input Type:** String **Description:** `RSName` is the recordset name.
Comment	
The method is not synchronized and executes in a separate thread.	

RSMovePrev

Method: RSMovePrev	
Method description: `RSMovePrev` moves the recordset cursor to the previous position. **JavaScript call:** `void RSMovePrev(string RSName)`	
Input	
`RSName`	**Input Type:** String **Description:** `RSName` is the recordset name.
Comment	
The method is not synchronized and executes in a separate thread.	

RSMoveTo

Method: RSMoveTo	
Method description: `RSMoveTo` moves the recordset cursor to the desired position. **JavaScript call:** `void RSMoveTo(string RSName, int nIndex)`	
Input	
`RSName`	**Input Type:** String **Description:** `RSName` is the recordset name.
`nIndex`	**Input Type:** Integer **Description:** `nIndex` is the desired cursor position.
Comment	
The method is not synchronized and executes in a separate thread.	

RSRefresh

Method: RSRefresh	
Method description: `RSRefresh` refreshes the recordset. The query is re-executed to get the new rows for the given recordset. **JavaScript call:** `void RSRefresh(string RSName)`	
Input	
`RSName`	**Input Type:** String **Description:** `RSName` is the recordset name.
Comment	
The method is not synchronized and executes in a separate thread.	

RSUpdate

Method: RSUpdate	
Method description: `RSUpdate` updates the recordset at the current cursor position. **JavaScript call:** `void RSUpdate(string RSName)`	
Input	
`RSName`	**Input Type:** String **Description:** `RSName` is the recordset name.
Comment	
The method is not synchronized and executes in a separate thread.	

Methods pertaining to form actions

This section outlines methods pertaining to form actions. For each method there is a description of the method, its method call (written in JavaScript), its parameters, and comments (where applicable). These methods are as follows (in alphabetical order):

- *CallDotNetCode*
- *CallUserAction*
- *CallWebServiceAction*
- *DoFormSubmit*
- *FillGroupMembers*

CallDotNetCode

Method: CallDotNetCode	
Method description: `CallDotNetCode` executes a specified .Net Code action. **JavaScript call:** `void CallDotNetCode(string strActionName)`	
Input	
`strActionName`	**Input Type:** String **Description:** `strActionName` is the name of the .Net Code action.

CallUserAction

Method: CallUserAction	
Method description: `CallUserAction` executes the .Net Code action. **JavaScript call:** `void CallUserAction(string EventName, string strActionName, string CPHName, string ActionID)`	
Input	
`EventName`	**Input Type:** String **Description:** `EventName` is the name of the event (such as "click").
`strActionName`	**Input Type:** String **Description:** `strActionName` is the name of the .Net Code action.
`CPHName`	**Input Type:** String **Description:** `CPHName` of the control on which event the action is set.
`ActionID`	**Input Type:** String **Description:** `ActionID` is an empty string.

CallWebServiceAction

Method: CallWebServiceAction	
Method description: `CallWebServiceAction` executes a specified Web service action. **JavaScript call:** `void CallWebServiceAction(string EventName, string strActionName, string CPHName)`	
Input	
`EventName`	**Input Type:** String **Description:** `EventName` is the name of the event (such as "click").
`strActionName`	**Input Type:** String **Description:** `strActionName` is the name of the Web service action.
`CPHName`	**Input Type:** String **Description:** `CPHName` of the control on which event the action is set.

DoFormSubmit

Method: DoFormSubmit
Method description: `DoFormSubmit` submits the task. **JavaScript call:** `void DoFormSubmit(short nSend)`
Comment
If the parameter is zero or less than zero then `DoFormSubmit` returns the form; otherwise `DoFormSubmit` submits the form.

FillGroupMembers

Method: FillGroupMembers	
Method description: `FillGroupMembers` fills the group members in a specified XML element. **JavaScript call:** `void FillGroupMembers(string strEventName, string strActionName, string strGroupName, string strVarName)`	
Input	
`EventName`	**Input Type:** String **Description:** `EventName` is the name of the event (such as "click").
`strActionName`	**Input Type:** String **Description:** `strActionName` is the name of the action.
`strGroupName`	**Input Type:** String **Description:** `strGroupName` is the group name.
`strVarName`	**Input Type:** String **Description:** `strVarName` XML element path in the group members to be filled.

Methods to manipulate XML schema

This section outlines methods which manipulate XML schema. For each method there is a description of the method, its method call (written in JavaScript), and its parameters. These methods are as follows (in alphabetical order):

- *XSAddNew*
- *XSClear*
- *XSDelete*
- *XSMoveFirst*
- *XSMoveLast*
- *XSMoveNext*
- *XSMovePrev*
- *XSMoveTo*
- *XSRefresh*
- *XSUpdate*

XSAddNew

Method: XSAddNew	
Method description: `XSAddNew` adds a new row in the complex repeated enumerator. The method returns the xpath of the newly added row. **JavaScript call:** `string XSAddNew(string strRepeatedNode)`	
Input	
`strRepeatedNode`	**Input Type:** String **Description:** `strRepeatedNode` is the dot (.) separated path of the complex repeated type.

XSClear

Method: XSClear	
Method description: `XSClear` clears the controls values linked with the complex repeated child elements. **JavaScript call:** `void XSClear(string strRepeatedNode)`	
Input	
`strRepeatedNode`	**Input Type:** String **Description:** `strRepeatedNode` is the dot (.) separated path of the complex repeated type.

XSDelete

Method: XSDelete	
Method description: `XSDelete` deletes the current row in the complex repeated enumerator. The method returns the xpath of the enumerator's current position. **JavaScript call:** `string XSDelete(string strRepeatedNode)`	
Input	
`strRepeatedNode`	**Input Type:** String **Description:** `strRepeatedNode` is the dot (.) separated path of the complex repeated type.

XSMoveFirst

Method: XSMoveFirst	
Method description: `XSMoveFirst` moves the complex repeated enumerator to the first position. The method returns the xpath to which the enumerator is moved. **JavaScript call:** `string XSMoveFirst(string strRepeatedNode)`	
Input	
`strRepeatedNode`	**Input Type:** String **Description:** `strRepeatedNode` is the dot (.) separated path of the complex repeated type.

XSMoveLast

Method: XSMoveLast	
Method description: `XSMoveLast` moves the complex repeated enumerator to the last position. The method returns the xpath to which the enumerator is moved. **JavaScript call:** `string XSMoveLast(string strRepeatedNode)`	
Input	
`strRepeatedNode`	**Input Type:** String **Description:** `strRepeatedNode` is the dot (.) separated path of the complex repeated type.

XSMoveNext

Method: XSMoveNext	
Method description: `XSMoveNext` moves the complex repeated enumerator to the next position. `strRepeatedNode` is the dot (.) separated path of the complex repeated type. The method returns the xpath to which the enumerator is moved. **JavaScript call:** `string XSMoveNext(string strRepeatedNode)`	
Input	
`strRepeatedNode`	**Input Type:** String **Description:** `strRepeatedNode` is the dot (.) separated path of the complex repeated type.

XSMovePrev

Method: XSMovePrev	
Method description: `XSMovePrev` moves the complex repeated enumerator to the previous position. The method returns the xpath to which the enumerator is moved. **JavaScript call:** `string XSMovePrev(string strRepeatedNode)`	
Input	
`strRepeatedNode`	**Input Type:** String **Description:** `strRepeatedNode` is the dot (.) separated path of the complex repeated type.

XSMoveTo

Method: XSMoveTo	
Method description: `XSMoveTo` moves the complex repeated enumerator to the position mentioned in the given xpath. **JavaScript call:** `string XSMoveTo(string strXpath)`	
Input	
`strXpath`	**Input Type:** String **Description:** `strXpath` is the xpath to which the enumerator is to be moved.

XSRefresh

Method: XSRefresh	
Method description: `XSRefresh` refreshes the complex repeated enumerator and moves it to the first position. **JavaScript call:** `void XSRefresh(string strRepeatedNode)`	
Input	
`strRepeatedNode`	**Input Type:** String **Description:** `strRepeatedNode` is the dot (.) separated path of the complex repeated type.

XSUpdate

Method: XSUpdate	
Method description: `XSUpdate` updates the data in the complex repeated enumerator at the current position. **JavaScript call:** `void XSUpdate(string strRepeatedNode)`	
Input	
`strRepeatedNode`	**Input Type:** String **Description:** `strRepeatedNode` is the dot (.) separated path of the complex repeated type.

Methods to manipulate controls

This section outlines methods which manipulate controls. For each method there is a description of the method, its method call (written in JavaScript), its parameters, and comments (where applicable). These methods are as follows (in alphabetical order):

- *GetControlDestinationValue*
- *GetControlSourceValue*
- *GetDataset*
- *IsOkToSwitchPage*
- *SetControlDestinationValue*
- *SwitchPage*

GetControlDestinationValue

Method: GetControlDestinationValue	
Method description: `GetControlDestinationValue` returns the unformatted value of the control's destination element. **JavaScript call:** `object GetControlDestinationValue(string strCtrlName)`	
Input	
`strCtrlName`	**Input Type:** String **Description:** `strCtrlName` is the name of the control.

GetControlSourceValue

Method: GetControlSourceValue	
Method description: `GetControlSourceValue` returns the unformatted value of the control's source. **JavaScript call:** `object GetControlSourceValue(string strCtrlName)`	
Input	
`strCtrlName`	**Input Type:** String **Description:** `strCtrlName` is the name of the control.

GetDataset

Method: GetDataset	
Method description: `GetDataset` returns the dataset if `strCPHName` is the name of schema grid control. **JavaScript call:** `object GetDataset(string strPageName, string strCPHName)`	
Input	
`strPageName`	**Input Type:** String **Description:** `strPageName` is the name of the page on which the Schema Grid control is hosted.
`strCPHName`	**Input Type:** String **Description:** `strCPHName` is the name of the Schema Grid control.
Comment	
`GetDataset` is used if the Schema Grid control is being manipulated through scripts; it fetches the dataset object from FCO through this method and passes that dataset object to the Schema Grid control interface method `SetData`.	

IsOkToSwitchPage

Method: IsOkToSwitchPage
Method description: `IsOkToSwitchPage` verifies whether it is safe to switch pages within an Ultimus Form. `IsOkToSwitchPage` returns `TRUE` if the form is not busy in any critical actions that can be affected by a page load. If the page should not be switched, `IsOkToSwitchPage` returns `FALSE`. **JavaScript call:** `bool IsOkToSwitchPage()`
Comment
As a best practice, this call should precede any `SwitchPage` calls.

SetControlDestinationValue

Method: SetControlDestinationValue	
Method description: `SetControlDestinationValue` sets the value in the control's destination element. **JavaScript call:** `void SetControlDestinationValue(string strCtrlName, object ObjValue)`	
Input	
`strCtrlName`	**Input Type:** String **Description:** `strCtrlName` is the name of the control.
`ObjValue`	**Input Type:** Object **Description:** `ObjValue` is the value to be set for the destination element.

SwitchPage

Method: SwitchPage	
Method description: `SwitchPage` loads the page specified in the `PageName` parameter. **JavaScript call:** `SwitchPage(string PageName)`	
Input	
`PageName`	**Input Type:** String **Description:** `PageName` is the name of the page to be loaded.
Comment	
`SwitchPage` works for both page name and Page ID.	

Extending the functionality of the Ultimus Database Grid and Schema Grid controls

When utilizing Ultimus Database Grid or Schema Grid controls in Ultimus Forms, methods outlined in the following sections may be used to manage data flow between the grid control(s) and the business process XML schema:

- *Grid control methods used with Database Grid controls*
- *Grid control methods used with Schema Grid controls*

Grid control methods used with Database Grid controls

This section outlines methods used with Database Grid controls. For each method there is a description of the method, its method call (written in JavaScript), its parameters, comments (where applicable), and an example. These methods are as follows (in alphabetical order):

- *AddRow*
- *DeleteRow*
- *GetColumnType*
- *GetColumnValue*
- *GetCurrentRow*
- *GetRowCount*
- *SetColumnValue*
- *SetCurrentRow*

- *SyncWithDB*
- *SyncWithSchema*

AddRow

Method: AddRow
Method description: `AddRow` adds a new row in the grid. The newly added row becomes the currently selected row. This method returns the index of the newly added row for further use. **JavaScript method call:** `public int AddRow()`
Example

`// This code snippet is written in JavaScript`
`document.DBGrid0.AddRow();`
`// Where DBGrid0 is the name of the Grid control.`

DeleteRow

Method: DeleteRow	
Method description: `DeleteRow` deletes the selected row from the grid. **JavaScript method call:** `public void DeleteRow(int nIndex)`	
Input	
`nIndex`	**Input Type:** Int **Description:** `nIndex` represents the index of the row to be deleted.
Example	

`// This code snippet is written in JavaScript`
`document.DBGrid0.DeleteRow(rowIndex);`
`// Where DBGrid0 is the name of the Grid control.`	

GetColumnType

Method: GetColumnType	
Method description: `GetColumnType` returns the type of a given recordset column. This method returns the column type as a string. **JavaScript method call:** `public string GetColumnType(string strColName)`	
Input	
`strColName`	**Input Type:** String **Description:** `strColName` represents the name of the column whose type is to be determined.
Example	

`// This code snippet is written in JavaScript`
`document.DBGrid0.GetColumnType('Column Name');`
`// Where DBGrid0 is the name of the Grid control and Column Name is the name of the column type to be returned.`	

GetColumnValue

Method: GetColumnValue	
Method description: `GetColumnValue` returns the value of a specified column of a specified grid row. This method returns the value of the specified column. **JavaScript method call:** `public object GetColumnValue(int nRowIndex, string strColName)`	
Input	
`nRowIndex`	**Input Type:** Int **Description:** `nRowIndex` represents the row whose column value is to be retrieved.
`strColName`	**Input Type:** String **Description:** `strColName` represents represents the column name whose value is to be retrieved.
Example	

`// This code snippet is written in JavaScript`
`document.DBGrid0.GetColumnValue(rowIndex,'Column Name');`
`// Where DBGrid0 is the name of the Grid control and Column Name is the name of the column value to be retrieved.`	

GetCurrentRow

Method: GetCurrentRow
Method description: `GetCurrentRow` retrieves the index of the currently selected row. This method returns the row index of the current row. If multiple rows are selected, then the index of the last row selected is returned. **JavaScript method call:** `public int GetCurrentRow()`
Example

`// This code snippet is written in JavaScript`
`document.DBGrid0.GetRowCount();`
`// Where DBGrid0 is the name of the Grid control.`

GetRowCount

Method: GetRowCount
Method description: `GetCurrentRow` retrieves the total number of rows in the grid. This method returns the integer value representing the number of rows in the grid. **JavaScript method call:** `public int GetRowCount()`
Example

`// This code snippet is written in JavaScript`
`document.DBGrid0.GetRowCount();`
`// Where DBGrid0 is the name of the Grid control.`

SetColumnValue

Method: SetColumnValue	
Method description: `SetColumnValue` sets the value of a specific column in the specified row in the grid. The value passed should be according to the type returned by *GetColumnType* (outlined on page 429). This method returns `TRUE` if `SetColumnvalue` is executed successfully; otherwise, this method returns `FALSE`. **JavaScript call:** `public bool SetColumnValue(int nRowIndex, string strColName, object objValue)`	
Input	
`nRowIndex`	**Input Type:** Int **Description:** `nRowIndex` represents represents the index of the row whose column value is to be set.
`strColName`	**Input Type:** String **Description:** `strColName` represents the name of the column for which the value is to be updated.
`objValue`	**Input Type:** Object **Description:** `objValue` represents the value to be set for the column.
Example	

`// This code snippet is written in JavaScript`
`document.DBGrid0.SetColumnValue(rowIndex,'Column Name','column value');`
 	
`// Where DBGrid0 is the name of the control, Column Name is the name of the column, and column value represents the value for the column.`	

SetCurrentRow

Method: SetCurrentRow	
Method description: `SetColumnRow` selects a specified row. If the provided row index is not available in the grid, then `FALSE` is returned and nothing is selected. This method returns `TRUE` if the row gets selected; otherwise, this method returns `FALSE`. **JavaScript method call:** `public bool SetCurrentRow(int nIndex)`	
Input	
`nIndex`	**Input Type:** Int **Description:** `nIndex` represents represents the row index for the selection of the row.

Example

```
// This code snippet is written in JavaScript
document.DBGrid0.SetCurrentRow(rowIndex);
// Where DBGrid0 is the name of the control.
```

SyncWithDB

Method: SyncWithDB
Method description: `SyncWithDB` refreshes the Dababase grid by retrieving updated values from the linked recordset and updates the value in the linked recordset that was updated in the grid after the previous call to `SyncWithDB`. This method returns `TRUE` if `SynWithDB` is executed successfully; otherwise, this method returns `FALSE`. **JavaScript method call:** `public bool SyncWithDB()`

Example

```
// This code snippet is written in JavaScript
document.DBGrid0.SyncWithDB();
// Where DBGrid0 is the name of the control.
```

SyncWithSchema

Method: SyncWithSchema
Method description: `SyncWithSchema` refreshes the schema by transferring the value of all selected rows. This method returns `TRUE` if `SynWIthSchema` is executed successfully; otherwise, this method returns `FALSE`. **JavaScript method call:** `public bool SyncSchema()`
Example

`// This code snippet is written in JavaScript`
`document.DBGrid0.SyncWithSchema();`
`// Where DBGrid0 is the name of the control.`

Grid control methods used with Schema Grid controls

This section outlines methods used with Schema Grid controls. For each method there is a description of the method, its method call (written in JavaScript), its parameters, comments (where applicable), and an example. These methods are as follows (in alphabetical order):

- *GetCurrentSelection*
- *RefreshData*
- *SetCurrentSelection*

GetCurrentSelection

Method: GetCurrentSelection
Method description: `GetCurrentSelection` retrieves the XPATH of the currently selected row. **Example 1:** If the schema element `OrderDate` of the second order for the third customer has been selected, then the XPATH is: `Customers[3].Orders[2]` **Example 2:** If the schema element `ItemID` of the fifth item of the third order for the fifth customer is selected, then the XPATH is: `Customers[5].Orders[3].Items[5]` **JavaScript method call:** `public string GetCurrentSelection()`
Example

`// This code snippet is written in JavaScript`
`document.SchemaGrid0.GetCurrentSelection();`
`// Where SchemaGrid0 is the name of the control.`

RefreshData

Method: RefreshData
Method description: `RefreshData` refreshes the schema grid with the linked schema element in the form schema. **JavaScript method call:** `public void RefreshData()`
Example
`// This code snippet is written in JavaScript` `document.SchemaGrid0.RefreshData();`

SetCurrentSelection

Method: SetCurrentSelection	
Method description: `SetCurrentSelection` selects a specific row in the grid. **JavaScript method call:** `public void SetCurrentSelection(string strXPath)`	
Input	
`strXPath`	**Input Type:** String **Description:** `strXPath` represents the XPath of the row to be selected.
Example	

`// This code snippet is written in JavaScript`
`document.SchemaGrid0.SetCurrentSelection('strXPath');`
`// Where SchemaGrid0 is the name of the control.`	

Inserting a Place Holder control

Ultimus Forms use .NET Code to provide flexibility in designing intelligent and powerful user interfaces. This capability may be further enhanced by using custom third-party controls via the Place Holder control. The Place Holder control may be used to execute a script when the control is utilized, thereby implementing unique functionality and capabilities not offered by Ultimus.

These custom controls can not only interact with the local step XML schema associated with the Ultimus Form, but may also interact with other controls through the JavaScript specified as a part of the controls. Local XML schema elements may be passed as arguments to the controls or receive data back from the controls.

The ability to develop the controls in .Net, Visual Basic, C, or Java provides practically unlimited functionality for expanding the Ultimus user interface.

Follow these steps to insert a custom third-party control using the Place Holder control:

1. In Ultimus BPM Studio, open the Ultimus Form of a step in which to place a custom third-party control.
2. Click on the Place Holder control () in the Toolbox, then click on the form where the Place Holder control is to display.

3. To configure the properties for the Place Holder control, right-click the Place Holder control box, then select **Properties**. The **Properties** window displays (as shown in Figure 28).

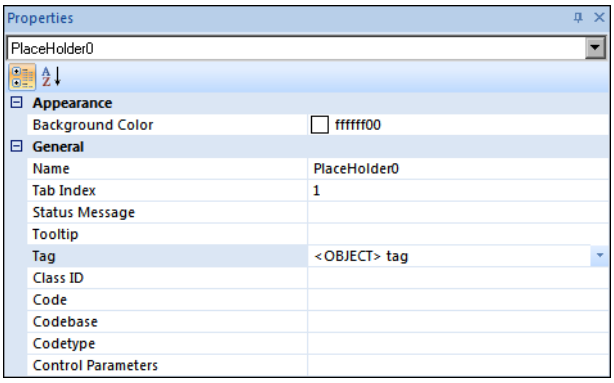

Figure 28. The Properties for a Place Holder control

4. From the **Tag** combo box, specify whether the third-party control is an object tag or a tag.
5. From the **Class ID** text box, specify the class ID value for the third-party control.
6. From the **Code** text box, specify the code for the third-party control.
7. From the **Codebase** text box, specify where to find the code for the third-party control.
8. From the **CodeType** text box, specify the code's Internet media type referred to in the **Code** text box.
9. To configure custom named-values to be passed to the third-party control, click within the **Control Parameters** field, then select the browse button to access the **Params List** dialog box (shown in Figure 29).

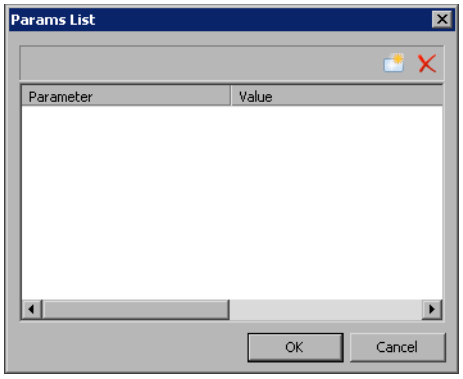

Figure 29. The Params List dialog box

For more specific information how to configure a Place Holder control, refer to *Place Holder* section in *Ultimus BPM Studio Help*.

The Form Designer generates shells for each object. These shells can be accessed from the **Scripts** category within the **Control Links** window. These four objects that Form Designer generates shells for are described below (in alphabetical order):

- *GetValue*
- *InitProps*
- *SetValue*
- *SetValueFromSS*

GetValue

Function: GetValue
Function description: The user must update the linked schema element in the XML step schema by the value of the control.
Example

```
// This code snippet is written in JavaScript
function ObjectName_GetValue()
{
// TODO:fill in body of this function to update the linked schema element in the FCO from the value in the control
// Linked to source : , destination :
}
```

InitProps

Function: InitProps
Function description: This function sets the properties of the object that cannot be set by the style statement. The user can fill in the function to set any properties as needed.
Example

```
// This code snippet is written in JavaScript
function ObjectName_InitProps()
{
// TODO:fill in body of this function to initialize the Control properties
}
```

SetValue

Function: SetValue
Function description: `setValue` can be any method in the control which "sets" the value of the control. This function is called when the control's value needs to be updated. Note that the FCO may only be referenced on pages by the first frame of the Web browser. Sub-forms may reference it as follows: `window.opener.document.theFCO`. The new value is passed in the "value" parameter, below. The user should call the appropriate method in the object to set its value.
Example

`// This code snippet is written in JavaScript`
`function ObjectName_SetValue(strValue)`
`{`
`// TODO:fill in body of this function to set the new value of the Control`
`}`

SetValueFromSS

Function: SetValueFromSS
Function description: This function is called when the value of the control needs to be refreshed from the FCO, usually when the page is loaded. `theFCO` is a global schema element in a page generated by the Forms Designer that points to the FCO. The user should get the new value from the FCO as shown and call the appropriate method in the object to set its value.
Example

`// This code snippet is written in JavaScript`
`function ObjectName_SetValueFromSS()`
`{`
`// TODO:fill in body of this function to refresh the Control values from the Form Control Object`
`// Linked to source : , destination :`
`}`

Example of a custom third-party control

If using an ActiveX control or a Javabean, here is an example of how these functions might be filled. The control name is MyCustomEdit.

```
// This code snippet is written in JavaScript
<!--Ultimus generated Script!-->
<SCRIPT LANGUAGE="JavaScript">
function MyCustomEdit_InitProps()
// setting additional properties for the control.
{
MyCustomEdit.setBorderStyle(5);
MyCustomEdit.setBackground(255, 255, 224);
MyCustomEdit.setForeground(52, 251, 152);
}
function MyCustomEdit_ SetValueFromSS ()
{
// The control has to get the data from element named MyVariable in
the step's XML
// schema. We call getVarValueControl passing the Control name and
XML element so
// it notifies when the Control value changes SetValue is a method
in the Control
// to change the Text in the control.
MyCustomEdit.setValue(theFCO.getVarValueControl("MyCustomEdit",
"TaskData.Global.MyVariable"));
}
function MyCustomEdit_SetValue(strValue)
{
// this is called when the value is to be updated in the control
MyCustomEdit.setValue(strValue);

function MyCustomEdit_GetValue()
{
// getValue() is the method to get the text in the control and pass
it to FCO to put
// in the element named MyVariable in step's XML schema.
}
theFCO.setVarValue("TaskData.Global.MyVariable", document.
MyCustomEdit.getValue());
```

```
</SCRIPT>
```

Providing an ActiveX wrapper to a .Net user control

If a third-party control is a .Net user control, then it works only at run time (not in Test or Simulation modes). To make the .Net user control function in all modes (TEST, SIMULATION, and REAL), an ActiveX wrapper must be placed on the .Net user control. An example is provided below:

```
[Guid("5AB3234B-0983-4c16-B164-6119B49B7DB2")]
public class MyCustomControl : UserControl,IMyInterface
{
    // Impelement Interface "IMyInterface"
    [ComRegisterFunction()]
    public static void RegisterClass(string key)
    {
        // Strip off HKEY_CLASSES_ROOT\ from the passed key
        StringBuilder sb = new StringBuilder(key);
        sb.Replace(@"HKEY_CLASSES_ROOT\", "");

        // Open the CLSID\{guid} key for write access
        RegistryKey k = Registry.ClassesRoot.OpenSubKey(sb.ToString(), true);

        // And create the 'Control' key - this allows it to show up in the ActiveX
        control container
        RegistryKey ctrl = k.CreateSubKey("Control");
        ctrl.Close();

        // Next create the CodeBase entry - needed if not string named and GACced.
        RegistryKey inprocServer32 = k.OpenSubKey("InprocServer32", true);
        inprocServer32.SetValue("CodeBase",Assembly.GetExecutingAssembly().CodeBase);
        inprocServer32.Close();

        // Finally close the main key
        k.Close();
    }

    [ComUnregisterFunction()]
    public static void UnregisterClass(string key)
    {
        StringBuilder sb = new StringBuilder(key);
        sb.Replace(@"HKEY_CLASSES_ROOT\", "");

        // Open HKCR\CLSID\{guid} for write access
        RegistryKey k = Registry.ClassesRoot.OpenSubKey(sb.ToString(), true);

        // Delete the 'Control' key, but don't throw an exception if it does not exist
        k.DeleteSubKey("Control", false);

        // Next open up InprocServer32
        RegistryKey inprocServer32 = k.OpenSubKey("InprocServer32", true);

        // And delete the CodeBase key, again not throwing if missing
```

```
            k.DeleteSubKey("CodeBase", false);
            // Finally close the main key
            k.Close();
        }
    }
```

This control must be registered with the `regasm MyCustomControl /codebase` command (where `MyCustomControl` is a binary name).

Furthermore, the class ID property for the Place Holder control must be clsid:uuid. In the example above, this property value would be `clsid:5AB3234B-0983-4c16-B164-6119B49B7DB2`.

Thereafter, the Place Holder control functions in all modes.

Distributing and using third-party controls

Custom third-party controls can be distributed even if a business process has been designed on a remote Ultimus BPM Studio Client computer.

To distribute a custom third-party control, follow these steps:

1. Copy the `OCX` file from the computer hosting remote Ultimus BPM Studio Client to the one hosting Ultimus BPM Server (into the *`systemdrive`*`\Inetpub\wwwroot\UltWeb` directory, where *`systemdrive`* is the drive Ultimus BPM Server is installed).
2. Register the `OCX` file on to the Ultimus BPM Server computer's Windows registry.
3. Open the business process in Ultimus BPM Studio Client on the computer hosting Ultimus BPM Server.
4. Specify a class path in the properties of the Place Holder control that is relative to the `UltWeb` directory on the computer hosting Ultimus BPM Server.
5. Publish the business process.
6. The first time the form is loaded, a prompt is displayed to download the Ultimus Place Holder control.

11

Using JavaScript in Ultimus Forms

This section discusses how to use JavaScript in Ultimus Forms. Using JavaScript in Ultimus Forms lets the process designer modify the functionality of the forms according to process requirements. This adds great flexibility to Ultimus Adaptive BPM Suite functionality.

Below is a sample of JavaScript and how to utilize it in Ultimus Adaptive BPM Suite 8.3 SP2:

1. Open the business process for editing in Ultimus BPM Studio.
2. View the Ultimus Form in which to use the JavaScript.
3. Select the **Form Actions** button from the Designer toolbar. The **Actions** window displays (as shown in Figure 30).

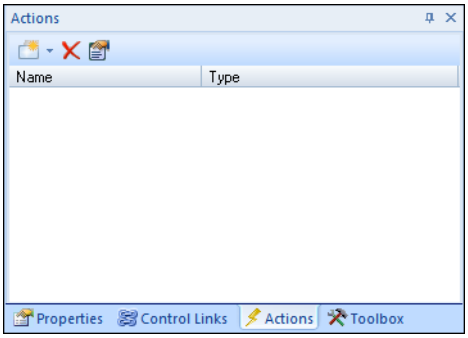

Figure 30. The Actions window

4. From the **New Action** button, select the **Execute Specified Script** option. The **Execute Specified Script** dialog box displays (as shown in Figure 31).

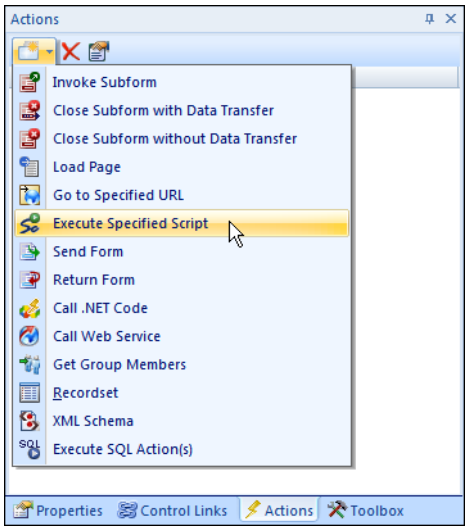

Figure 31. Selecting the Execute Specified Script option

The **Execute Specified Script** dialog box appears (as shown in Figure 32).

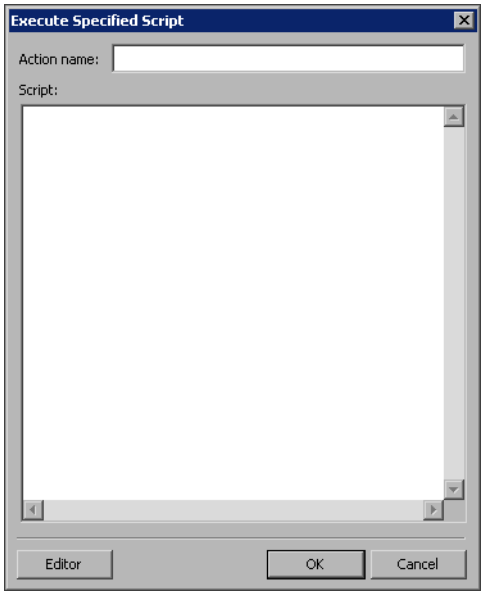

Figure 32. The Execute Specified Script dialog box

5. Author the required script in the script window. For this example, use the following JavaScript:

```
window.parent.frames(0).document.theFCO.setVarValue("TaskData.Global.MyVariable"
, "This is the value to be set");
```

6. Click the **OK** button.

Once the actions are saved they can be linked with other controls. For example, to link the action with a Push Button control, the following steps (in overview) would be followed:

a. Place the Push Button control on the Ultimus Form.

b. Configure the control links for the Push Button control. Within the **Event Actions** category, specify an event to associate with the JavaScript action.

12

Error status codes returned by Flobots

When training an Ultimus Flobot for an action, an XML schema element may be specified to record the error status code that is returned. After a Flobot has completed its work, an integer value is returned to the schema element.

Below is a list of integer values and their definitions for each Ultimus Flobot that is bundled with Ultimus Adaptive BPM Suite 8.3 SP2.

The following table outlines the returned error codes for the .NET Code Flobot.

Table 27. .NET Code Flobot returned error codes

Error Code	Error Description
0	Successful completion
1	Failure
10	Action not executed; enable evaluated to FALSE

The following table outlines the returned error codes for the ASCII Flobot.

Table 28. ASCII Flobot returned error codes

Error Code	Error Description
0	Successful completion
1	Missing output file name
2	Invalid path or write permission failed
3	Row count invalid
9	Failed for an unknown reason
10	Action not executed; enable evaluated as FALSE

The following table outlines the returned error codes for the Database Flobot.

Table 29. Database Flobot returned error codes

Error Code	Error Description
0	Successful completion
1	Could not set up connection
2	Schema element not found
3	Could not add to database
4	Could not execute query
5	Failed to update schema with data
9	Failed for an unknown reason
10	Action not executed; enable evaluated as `FALSE`

The following table outlines the returned error codes for the E-mail Flobot.

Table 30. E-mail Flobot returned error codes

Error Code	Error Description
0	Successful completion
1	Could not initialize mail component
2	Could not log on
3	Could not send e-mail(s)
4	Invalid connector specified
5	Could not find at least one specified attachment
9	Failed for an unknown reason
10	Action not executed; enable evaluated as `FALSE`

The following table outlines the returned error codes for the Excel Flobot.

Table 31. Excel Flobot returned error codes

Error Code	Error Description
0	Successful completion
1	Action failed
3	Unable to print document
5	Unable to run macro
6	Unable to save
7	Unable to save file (file name cannot be empty)
8	Unable to start Excel application
9	Unable to transfer data
10	Action not executed; enable evaluated as `FALSE`
11	Unable to print document (printer name cannot be empty)
12	Unable to run macro (macro name cannot be empty)
13	Unable to save (specified extension is not supported)
14	Unable to save (specified directory does not exist)
15	Unable to save (the user does not have required permissions)
16	Unable to save (the file path specified is too long; it must be less than 248 characters
17	Unable to save (the specified file path is invalid)

The following table outlines the returned error codes for the Exchange Flobot.

Table 32. Exchange Flobot returned error codes

Error Code	Error Description
0	Successful completion
1	No recipient(s) could be resolved
2	Date invalid for task
3	Date or time invalid for event
4	Could not connect to profile

Table 32. Exchange Flobot returned error codes *(Continued)*

Error Code	Error Description
5	Could not attach files
9	Failed for an unknown reason
10	Action not executed; enable evaluated as FALSE

The following table outlines the returned error codes for the SharePoint Flobot.

Table 33. SharePoint Flobot returned error codes

Error Code	Error Description
0	Successful completion
1	Could not execute action
2	Operation specified on invalid file(s)
3	Could not copy file(s) on network or local path
4	Could not get file(s) from SharePoint
5	Could not upload or move file(s) on SharePoint
6	Could not delete file(s)
7	Failed to create object
8	Failed to delete object
9	Failed for an unknown reason
10	Action not executed; enable evaluated as FALSE
11	No user could be added
12	Could not retrieve metadata from SharePoint
13	Could not write metadata to SharePoint
14	Could not retrieve attachment

The following table outlines the returned error codes for the Web Services Flobot.

Table 34. Web Services Flobot returned error codes

Error Code	Error Description
0	Successful completion
1	Failure
10	Action not executed; enable evaluated as FALSE

The following table outlines the returned error codes for the Word Flobot.

Table 35. Word Flobot returned error codes

Error Code	Error Description
0	Successful completion
1	Failure
3	Unable to print document
5	Unable to run macro
6	Unable to save
7	Unable to save file (file name cannot be empty)
8	Unable to start Word application
9	Unable to transfer data
10	Action not executed; enable evaluated as FALSE
11	Unable to print document (printer name cannot be empty)
12	Unable to run macro (macro name cannot be empty)
13	Unable to save (specified extension is not supported)
14	Unable to save (specified directory does not exist)
15	Unable to save (the user does not have required permissions)
16	Unable to save (the file path specified is too long; it must be less than 248 characters
17	Unable to save (the specified file path is invalid)

The following table outlines the returned error codes for the XML Flobot.

Table 36. XML Flobot returned error codes

Error Codes	Error Description
0	Successful completion
1	Could not transfer XML (data transfer itself could not take place)
3	Could not write output XML (data transfer occurred but the resulting XML could not be written)
4	Could not update transferred XML (on Ultimus' side only, transferred XML data did not reflect in Ultimus data)
5	XML file path missing or invalid
6	XML file specified does not conform to the given XML schema
9	Failed for an unknown reason
10	Action not executed; enable evaluated as `FALSE`

Index

A

C

D

E

F

G

I

J

L

M

O

P

Q

R

S

T

U

V

X