Remote API Guide

AgilePoint BPMS v5.0 SP2

Document Revision r5.2.5

November 2011

Contents

Preface

Disclaimer of Warranty

AgilePoint, Inc. makes no representations or warranties, either express or implied, by or with respect to anything in this document, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose or for any indirect, special or consequential damages.

Copyright

Trademarks

AgilePoint, Inc. and AgilePoint's products are trademarks of AgilePoint Inc. References to other companies and their products use trademarks owned by the respective companies and are for reference purpose only.

Government Rights Legend

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the applicable license agreement and as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (Oct 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14, as applicable.

Virus-free software policy

AgilePoint recognizes that viruses are a significant security consideration for our customers. To date, we have had no report of AgilePoint BPMS carries any virus. AgilePoint takes the following measures to ensure our software is free of viruses upon delivery:

- AgilePoint is built on top of Microsoft .NET framework. The pre-compiled executable is a.NET Common Language Runtime (CLR) application, not a native machine binary. As far as is known at this time, there are no viruses that infect .NET CLR executables.
- The virtual environment for the product packaging process in is fully isolated and protected, and anti-virus software is installed and running during packaging.
- The deliverable package is scanned by anti-virus software before upload to our customer download site.

Document Revision Numbers

AgilePoint documentation uses the revision number format **r*X.Y.Z***. The letters and numbers in this revision number can be interpreted as follows:

- **r** - Indicates "revision." This helps to differentiate the document *version* numbers, which start with **v**.
- ***X*** - The major version number for AgilePoint BPMS to which this document refers. For example, AgilePoint releases 5.0, 5.0 SP1, and 5.5 would all have an ***X*** value of **5**.
- ***Y*** - The major document revision number. This number typically changes only when either there is a new AgilePoint release, or there are major changes to the document.
- ***Z*** - The minor document revision number. This number is incremented each time the document is republished.

AgilePoint Documentation in PDF and HTML

AgilePoint documentation is provided in both print-friendly (PDF) and web-based (HTML) formats.

Advantages of HTML Documentation

- HTML is the **primary delivery format** for AgilePoint documentation.
- Unified, global **search** across all documentation. PDF documents allow you to search only within the context of a given PDF file.
- **All hyperlinks supported**. Links in PDFs are only supported in certain contexts.
- "One-stop shopping" for all information related to AgilePoint BPMS.
- The HTML documentation is updated more frequently than the PDF documentation. Web-based documentation is updated periodically between AgilePoint releases to address errors and omissions, but the PDF documentation is updated only at the time of a software release.

Advantages of PDF Documentation

PDFs can be more easily **printed**, **archived**, and **transferred** (such as by FTP or email) than HTML documentation.

For more information, see Downloading Files and Sharing Links from the Documentation Library on the AgilePoint Support Portal.

Contacting AgilePoint Sales

AgilePoint is a leading Business Process Management System (BPMS) provider created by a team of driven people who strive to incorporate the principles of relentless innovation for the benefit of our customers. Our mission is to help companies of any size attain and sustain operational success through process excellence.

Headquarters: AgilePoint Corporation 1916C Old Middlefield Way Mountain View, CA 94043, USA

Tel: (650) 968 - 6789

Fax: (650) 968 - 6785

Email: info@agilepoint.com

Web site: www.agilepoint.com

International: For AgilePoint EMEA and AgilePoint Asia Pacific, please call the AgilePoint Corporate Office for contact information.

Contacting Customer Support

To contact AgilePoint Support, please submit a ticket on the AgilePoint Support Portal: http://support.agilepoint.com/SupportPortal/

If you do not have a Support Portal account, you can send an email to request one: support@agilepoint.com

AgilePoint API

This document describes the AgilePoint remote API, which you can access using a web service or a Windows service (WCF).

The AgilePoint remote API is designed and implemented using the Microsoft .Net web service framework, but you can view the WSDL using Internet Explorer by connecting AgilePoint Server at a URL the format:

- **Workflow API** - http://[qualified machine name]:[port]/[AgilePoint virtual directory]/workflow.asmx?WSDL
- **Administration API** - http://[qualified machine name]:[port]/[AgilePoint virtual directory]/admin.asmx?WSDL

WSDL is a web service standard that allows any client application to consume the AgilePoint API. However with .NET, Visual Studio or a command line can help to generate a web service proxy class that is used to consume the AgilePoint web service using the .NET object model, rather than HTTP and XML. With AgilePoint Developer, AgilePoint provides a prebuilt web service proxy to simplify the process even further. Using the AgilePoint web service proxy also helps with upgrade and deployment because updates will occur using the AgilePoint upgrade kit.

Basic Programming Tasks

This section describes the basic programming tasks that are required for the AgilePoint API:

- Authentication
- Adding a namespace and reference
- Catching exceptions
- Making calls

Code Examples

This section provides general code examples for the remote APIs. The code varies slightly depending upon whether AgilePoint Server is installed in IIS or Windows Service mode.

AgilePoint Server Running in IIS Mode

GetAdminService

```
public IWFAdminService GetAdminService()
{
    // initiated an object of proxy class of AgilePoint Web
 Service
    IWFAdminService svc = new AdminService("http://[machine]/
[virtual
        directory]");
    // set URL
    svc.CookieContainer = new System.Net.CookieContainer();
    // set Credentials
    svc.Credentials = new
 System.Net.NetworkCredential(userName, password,
        domainName);
    // or take default credential
    // svc.Credentials =
 System.Net.CredentialCache.DefaultCredentials;

    try
    {
        // set client application name
        svc.SetClientAppName("Samples");
        // set locale
        svc.SetClientLocale("en-US"); // AgilePoint support 13
 languages
        // check authenticated and return qualified user name
        string qualifiedDomainUserName =
 svc.CheckAuthenticated();
```

```
    }

    catch(Exception ex)
    {
        string error = ShUtil.GetSoapMessage(ex)
        //log/throw the exception
    }

    Finally
    {
        return svc;
    }
}
```

GetWorkflowService

```
public IWFWorkflowService GetWorkflowService ()
{
    // initiated an object of proxy class of AgilePoint Web
 Service
    IWFWorkflowService svc = new WorkflowService("http://
[machine]/[virtual
        directory]");

    // set URL
    svc.CookieContainer = new System.Net.CookieContainer();

    // set Credentials
    svc.Credentials = new
 System.Net.NetworkCredential(userName, password,
        domainName);

    // or take default credential
    // svc.Credentials =
 System.Net.CredentialCache.DefaultCredentials;
    try
    {
        // set client application name
        svc.SetClientAppName("Samples");

        // set locale
        svc.SetClientLocale("en-US"); // AgilePoint supports
 13 languages

        // check authenticated and return qualified user name
        string qualifiedDomainUserName =
 svc.CheckAuthenticated();

    }

    catch(Exception ex)
    {
```

```
        string error = ShUtil.GetSoapMessage(ex)
        //log/throw the exception
    }

    Finally
    {
        return svc;
    }
}
```

AgilePoint Server Running in Windows Service Mode

GetAdminService

```
public IWFAdminService GetAdminService()
{
    string user = this.Context.User.Identity.Name;

    // Set Credentials – Windows Authentication
    System.Net.ICredentials credentials =
        System.Net.CredentialCache.DefaultCredentials;
    // In case of form authentication
    //System.Net.NetworkCredential credentials = new
        System.Net.NetworkCredential(userName, password,
 domain);

    string locale = "en-us";

    string adminBinding =

 (String)ConfigurationSettings.AppSettings["AdminBindingUsed"];
    IWFAdminService m_adm = new
 WCFAdminProxy("MyApplicationName",
        "", locale, user, credentials, adminBinding);

    Return m_adm;
}
```

GetWorkflowService

```
public IWFWorkflowService GetWorkflowService ()
{
    string user = this.Context.User.Identity.Name;

    // Set Credentials – Windows Authentication
    System.Net.ICredentials credentials =
        System.Net.CredentialCache.DefaultCredentials;
```

```
    // In case of form authentication
    //System.Net.NetworkCredential credentials = new
        System.Net.NetworkCredential(userName, password,
 domain);

    string locale = "en-us";

    string workFlowBinding =

 (String)ConfigurationSettings.AppSettings["WorkFlowBindingUsed"];
    IWFWorkflowService m_api = new
 WCFWorkflowProxy("MyApplicationName",
             "", locale, user, credentials, workFlowBinding);

    return m_api;
}
```

(Windows Service Installation Only) Run the Windows Service Client Utility Tool

If you are using AgilePoint Server in a Windows Service environment, you must run the Windows Service Client Utility Tool on your application configuration file to configure the binding with the AgilePoint Server.

Open the Client Utility Tool on Your Machine

On the AgilePoint Server machine, navigate to **[AgilePoint Server Installation]\SVCUtilityTool \AgilePointWindowsServiceClientUtilityTool.exe**

This is only installed on the AgilePoint Server machine.

Open the Client Utility Tool from the AgilePoint Installation Program

Open the AgilePoint installation Setup.exe file, and click **AgilePoint Utilities and Other Support Files > AgilePoint Windows Service Client Utility Tool**.

In the Client Utility Tool, open the **Help** link for detailed instructions.

Authentication

To communicate with the AgilePoint API, you must establish a session, which associates an authenticated user with a set of calls. The client must provide credentials to the AgilePoint Server using Windows authentication.

There are two main ways to provide credentials to the AgilePoint Server: Specify the user's credentials, or use the default credentials.

Specify the Credentials for a User

Specify the user name, password, and domain name for a user. The domain name could be a Windows Domain Name or a Local Host Name. The following example shows the syntax for establishing credentials in this way:

```
System.Net.ICredentials = new
    System.Net.NetworkCredential(userName, password, domainName);
```

Use the Default Credentials

Use the default credentials for a user. The default is the system credentials for the current security context in which the application is running. For a client-side application, these are usually the Windows credentials (user name, password, and domain) of the user running the application. For ASP.NET applications, the default credentials are the user credentials of the authenticated user, or the user being impersonated.

For more information, see Surrogate on the AgilePoint Support Portal.

```
System.Net.ICredentials =
    System.Net.CredentialCache.DefaultCredentials;
```

Namespace Reference

With an out-of-the-box web service proxy included with AgilePoint, a reference to the namespace must be created. There are two methods to create a namespace reference:

1. Use the AgilePoint pre-compiled web service proxy. This method is recommended because it allows for easier upgrades. If you use AgilePoint proxy, you do not need to change your references to it when you upgrade the AgilePoint server because the new functions are included with the AgilePoint proxy.
   a. Add the following commands to your assembly file:
      - Ascentn.workflow.shared
      - Ascentn.workflow.WFBase
      - Ascentn.workflow.WFXML
   b. Add a namespace using Ascentn.WorkflowBase
2. Use Microsoft Visual Studio to generate a web service proxy. The disadvantage to this method is that you must regenerate the proxy each time you upgrade the AgilePoint Server.

Catching Exceptions

The exception that a web service throws contains a lot of information, and most of it is not easy to read. AgilePoint Server tags the readable message for end-users.

Call the following function to extract the error message:

```
String error = ShUtil.GetSoapMessage(ex);
```

ex is the exception object that contains error message from AgilePoint Server

Making Calls

Calls within the AgilePoint API fall into two categories – synchronous or asynchronous.

Synchronous Call

Synchronous calls are used for short transactions. In a synchronous call, a request is sent to the AgilePoint Server, the server acknowledges the request, and then acts upon it immediately.

Asynchronous Call

Synchronous calls are used for short transactions. In a synchronous call, a request is sent to the AgilePoint Server, the server acknowledges the request, and then acts upon it immediately.

Asynchronous calls are used for longer transactions. In an asynchronous call, a request is sent to the AgilePoint Server, the server acknowledges the request, but it does not immediately act upon it immediately. The server creates a WFEvent object, which contains the call's status, and returns the WFEvent object to the client. The client can call GetEvent(EventID) to retrieve the WFEvent object with the updated status. The status can be:

- **Processed** – The transaction was completed successfully.
- **Failed** – The transaction failed, and was not completed.
- **Sent** – The call was received, but it has not been acted upon.

Completing an asynchronous call could take any amount of time, from one second to several days. As a best practice a user interface should handle GetEvent() calls to update end users or the application itself regarding the status of asynchronous calls. You might, for example, use Ajax to check status to display on an ASP.NET page in real time.

Methods

This section includes references for all methods within the AgilePoint Web Service API.

Common Methods

This section describes some commonly used methods.

Check Authentication

Description

This call is used to verify whether the specified user is a registered user on the AgilePoint Server.

Syntax

public virtual string CheckAuthenticated()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

If the user is a registered user, the qualified user name in the format of DomainName\UserName is returned. Otherwise, null is returned.

Example

```
//This example is for an ASP.net application.//
public static string Connect(
System.Web.SessionState.HttpSessionState session,
System.Net.ICredentials credential, string appName, string
 locale)
    {
    string url =
 System.Configuration.ConfigurationManager.AppSettings.Get("ServerUrl");
```

```
    WorkflowService svc = new WorkflowService(url);
    AdminService adm = new AdminService(url);
    System.Net.CookieContainer cookieContainer = new
    System.Net.CookieContainer();
    svc.Credentials = credential;
    svc.CookieContainer = cookieContainer;
    svc.SetClientAppName(appName);
    svc.SetClientLocale(locale);
    string userName = svc.CheckAuthenticated();
    adm.Credentials = credential;
    adm.CookieContainer = cookieContainer;
    adm.SetClientAppName(appName);
    adm.SetClientLocale(locale);

    //Assume the ASP.net is on session
    WFCommonPage.SetAdm(session, adm);
    WFCommonPage.SetAPI(session, api);

    //return fully qualified Domain username
    return userName;
    }
```

Supported Versions

3.2.0.4 and higher

Surrogate

Description

IIS does not support users who do not use Windows Active Directory authentication. To address this issue, AgilePoint uses a special type of user called an impersonator. Impersonators enable client applications to authenticate end users who use Active Directory authentication, as well as those who do not.

Impersonators must meet the following requirements:

1. The impersonator must be a Windows Active Directory user to pass IIS authentication.
2. The impersonator must be registered on the AgilePoint Server. The registration can be done through Enterprise Manager.
3. The impersonator must be registered for the application under the Extension of AgilePoint Server Configuration. The application name is case sensitive.
4. The impersonator does not need to be the administrator for AgilePoint Server, or even have workflow execution rights on AgilePoint.

This Surrogate function allows the impersonator to act as a surrogate for the specified user to complete IIS authentication. Once the authentication has passed, the web service API will be called based on the rights granted to the specified user on AgilePoint Server.

This function is called before calling any other AgilePoint Web Service API.

Syntax

public virtual void Surrogate(string userName)

Parameters

Name	Type	Description
userName	string	A string that contains the user name for the user. This member must be a registered AgilePoint user.

Output

None.

Example

```
public IWFWorkflowService GetWorkflowService(
System.Net.ICredentials credentials, string
surrogateUsername)
{
    IWFWorkflowService svc = GetAdm(credentials);
 svc.Surrogate(surrogateUsername);
 return svc;
}

//Web service using Impersonator credentials that is
 registered in the AgilePoint Server
//configuration with application name
public IWFWorkflowService
 GetWorkflowService(System.Net.ICredentials credentials)
{
 string url = … // AgilePoint Server web service Url
 System.Net.CookieContainer cookieContainer = new
 System.Net.CookieContainer();
 WFWorkflowService svc = new WFWorkflowService(url);
 svc.CookieContainer = cookieContainer;
 svc.Credentials = credentials;
 svc.SetClientAppName([your application name]);
 //Current Locale
 svc.SetClientLocale(Thread.CurrentThread.CurrentUICulture.Name);
 return svc;
}
```

Supported Versions

3.2.0.4 and higher

Surrogate With Application Name and Locale

Description

This function is similar to Surrogate, with the ability to set the application name and locale at the same time.Calling the functions SetClientAppName and SetClientLocale is not needed if this function is called.

Syntax

public virtual void Surrogate(string userName, string appName, string locale)

Parameters

Name	Type	Description
userName	string	A string that contains the user name for the user. This member must be a registered AgilePoint user.
appName	string	A string that contains the name of the application. The name is case-sensitive.
locale	string	A string that contains the client locale in the format en-US.

Output

None.

Example

```
public IWFWorkflowService
 GetWorkflowService(System.Net.ICredentials
credentials, string surrogateUsername, string appName, string
 locale)
 {
```

```
    IWFWorkflowService svc = GetWorkflowService(credentials);
    svc.Surrogate(surrogateUsername, surrogate appName,
 surrogate locale);
    return svc;
 }

public IWFAdminService
 GetWorkflowService(System.Net.ICredentials credentials)
 {
    //The body is the same as Surrogate
 }
```

Supported Versions

3.2.0.4 and higher

Set Client AppLication Name

Description

This call is used to set the current application name. The application can be a web/Windows/Windows service application that calls the AgilePoint server for business process actions.

Syntax

public virtual void SetClientAppName(string appName)

Parameters

Name	Type	Description
appName	string	A string that contains the name of the application. The name is case-sensitive.

Output

None.

Example

```
See previous example
```

Supported Versions

3.2.0.4 and higher

Set Client Locale

Description

This call is used to set the locale for the client application that calls AgilePoint Server. A client application can be a web/Windows/Windows service application.

Syntax

public virtual void SetClientLocale(String locale)

Parameters

Name	Type	Description
locale	string	A string that contains the client locale in the format en-US.

Output

None.

Example

```
None.
```

Supported Versions

3.2.0.4 and higher

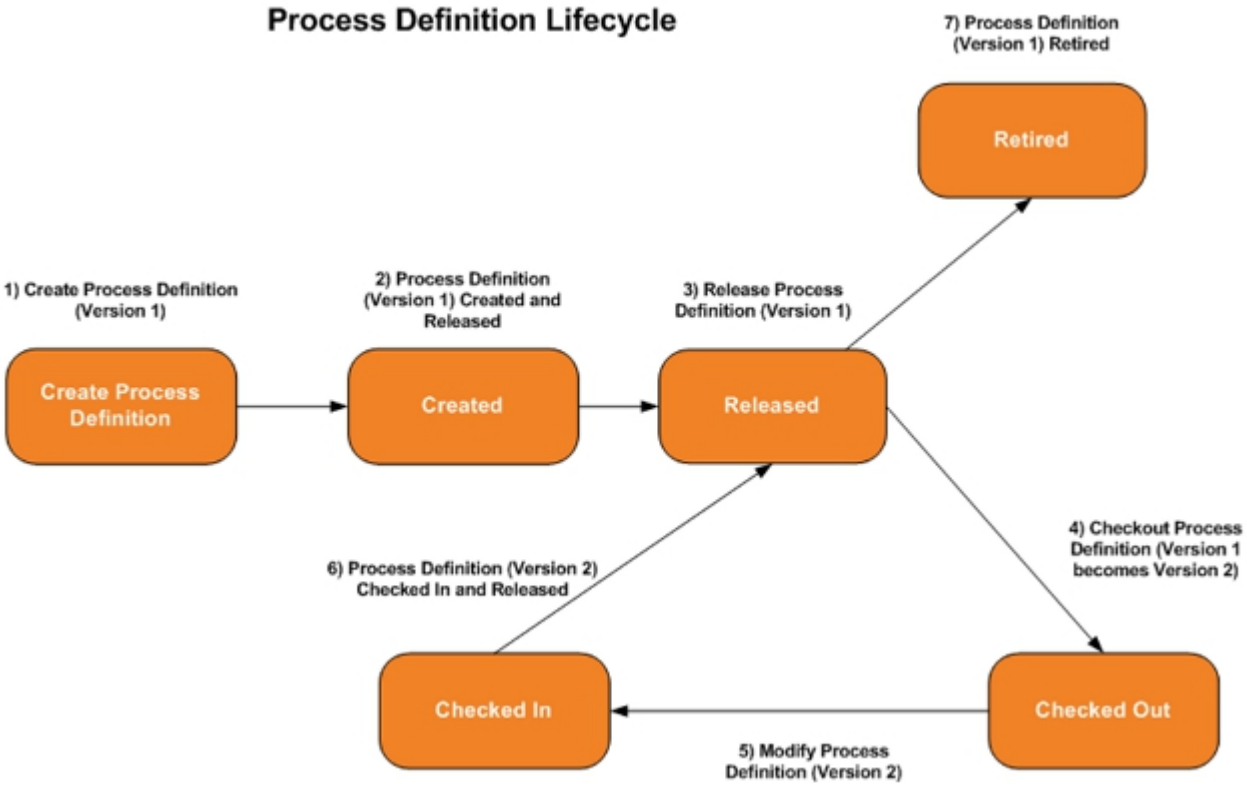
Process Definition Methods

This section describes how process definitions are managed. The AgilePoint server supports version control to ensure that process definitions are archived and maintained for later use. A process

definition can spawn many process instances. Employing version controls allows the AgilePoint Server to store multiple versions of a single process definition.

The project lifecycle of a process definition can be defined by the following steps:

1. **Create Process Definition** – Create a process definition. The version for the initial process definition is version 1.
2. **Created** –Confirm that the process definition has been created.
3. **Release process definition** – Once a process definition has been created, it must be "released." This allows the process definition to be checked out and edited/modified.
4. **Check out process definition** – If changes or modifications are made, the process definition is "checked out." Changes can then be made to the process definition.
5. **Modify process definition** – After a process definition has been checked out, changes can be made to the process definition.
6. **Check in process definition** – Once all changes/modifications have been made, the process definition is checked in, so the changes are committed to the process definition. The process definition is now version 2.
7. **Retire process definition** – Once version 2 of the process definition has been checked in, the previous version 1 is retired. Retired process definitions can also be deleted.



Create Process Definition

Description

Adds a new process definition to the AgilePoint Server.

Syntax

public virtual string CreateProcDef(string xml)

Parameters

Name	Type	Description
xml	string	A string that contains process definition in XML format. To generate the process definition file in XML format, in AgilePoint Envision, click **File > Export & Import > Save As Deploying File(xml)**. You can also download the process definition XML from AgilePoint Enterprise Manager.

Output

Unique ID of the process definition, which the AgilePoint system generates.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string xml = …// Process definition XML as string

if( set release date )
 {
    WFProcessDefinition pd = new WFProcessDefinition();
    GraphicImage g = new GraphicImage();
    ProcDefXmlParser parser = new ProcDefXmlParser(new
    WFDefaultActivityInstantiator(), pd, g);
    parser.Parse(xml);
    pd.ReleaseDate = DateTime.Now; //Set release date
    ProcDefXmlWriter w = new ProcDefXmlWriter(pd, g);
```

```
    xml = w.WriteTostring();
    string procDefID = svc.CreateProcDef(xml);
 }

else // not to release process template
 {
    string procDefID = svc.CreateProcDef(xml);
 }
```

Supported Versions

3.2.0.4 and higher

Release Process Definition

Description

Releases a process definition from the AgilePoint Server.

Syntax

public virtual void ReleaseProcDef(string processTemplateID)

Parameters

Name	Type	Description
processTemplateID	string	The unique identifier for the process definition to be checked out for modification.

Output

None.

Example

```
//Sample for using Workflow.ReleaseProcDef using Console
 Application

try
 {
    IWFWorkflowService svc = GetWorkflowService();
```

```
    string processDefinitionID = …
    svc.ReleaseProcDef(processDefinitionID);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " +
 ShUtil.GetSoapMessage( ex ) );
 }
```

Supported Versions

3.2.0.4 and higher

CheckOut Process Definition

Description

This method is used to manage process definition versioning by setting the process definition status to CheckedOut based on a given process definition ID. Only process definitions with the status of Released can transition into the CheckedOut status.

Syntax

public virtual string CheckoutProcDef(string pID)

Parameters

Name	Type	Description
processTemplateID	string	The unique identifier for the process definition to be checked out for modification.

Output

The process definition, in XML format, that has been checked out.

Example

```
IWFWorkflowService svc = GetWorkflowService();

try
```

```
 {
    string processDefinitionID = … // process definition to be
 checked out
    stringprocessDefinitionXML =
    svc.CheckoutProcDef(processDefinitionID);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " +
 ShUtil.GetSoapMessage( ex ) );
 }
```

Supported Versions

3.2.0.4 or higher

Uncheckout Process Definition

Description

Undoes a check-out for a process definition. This method returns the status of a process definition from CheckedOut to Released without making changes to the process definition, or changing the version number.

Syntax

public virtual void UnCheckOutProcDef(string pID)

Parameters

Name	Type	Description
processTemplateID	string	The unique identifier for the process definition to be checked out for modification.

Output

None.

Example

```
try
 {
 IWFWorkflowService svc = GetWorkflowService();
 string processTemplateID = …
 stringprocessDefinitionXML =
svc.UncheckoutProcDef(processTemplateID);
 }

catch (Exception ex)
 {
  Console.WriteLine("Message:\n" +
 ShUtil.GetSoapMessage( ex ) );
 }
```

Supported Versions

3.2.0.4 and higher

Check In Process Definition

Description

Checks in the process definition to the AgilePoint Server and returns the process definition identifier. This method accepts a string with the updated process definition in XML format.

Syntax

public virtual string CheckinProcDef(string xml)

Parameters

Name	Type	Description
xml	string	A string that contains process definition in XML format. To generate the process definition file in XML format, in AgilePoint Envision, click **File > Export & Import > Save As Deploying File(xml)**. You can also

Name	Type	Description
		download the process definition XML from AgilePoint Enterprise Manager.

Output

A new process definition ID.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string xml = …// Process definition XML as string
WFProcessDefinition pd = new WFProcessDefinition();
GraphicImage g = new GraphicImage();
ProcDefXmlParser parser = new ProcDefXmlParser(new
WFDefaultActivityInstantiator(), pd, g);
parser.Parse(xml);

if (release process definition immediately)
 {
    pd.ReleaseDate = DateTime.Now;
    pd.Version = .. // new version
    string procDefID = svc.CheckinProcDef( xml );
    svc.ReleaseProcDef(processDefinitionID);
 }

else if(release process definition at specific date in the
 future)
 {
    pd.ReleaseDate = …// a specific date in the future
    pd.Version = .. // new version
    string processDefinitionID = svc.CheckinProcDef( xml );
    svc.ReleaseProcDef(processDefinitionID);
    }

else // not release process definition
    {
    pd.ReleaseDate = Constants.NullDate;
    string processDefinitionID = svc.CheckinProcDef( xml );
 }
```

Supported Versions

3.2.0.4 or higher

Delete Process Definition

Description

Deletes the process definition and all of the process instances associated with the process definition. The process definition cannot be deleted if one or more process instances associated with the process definition is running or suspended. The function may take a long time to execute if there are many process instances associated with the process definition.

Syntax

public virtual void DeleteProcDef(string pID)

Parameters

Name	Type	Description
processTemplateID	string	The unique identifier for the process definition to be checked out for modification.

Output

None.

Example

```
IWFWorkflowService svc = GetWorkflowService();

string processTemplateID = ..// The unique identifier of the
 process definition to be deleted
svc.DeleteProcDef(processTemplateID);
```

Supported Versions

3.2.0.4 and higher

Get Base Process Definition ID

Description

Retrieves the ID for the first version of the process definition, called the base process definition. All subsequent process definition versions have the same base process definition ID. This call retrieves the base process definition ID with the specified process definition name.

Syntax

public virtual string GetBaseProcDefID(string pName)

Parameters

Name	Type	Description
processDefinitionName	string	The name of the process definition.

Output

string that contains the base process definition ID.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string processDefinitionName = "CreateRequest";
string baseProcessDefinitionID=
svc.GetBaseProcDefID(processDefinitionName);
```

Supported Versions

3.2.0.4 and higher

Get Process Definitions

Description

Retrieves all of process definition objects.

Syntax

public virtual WFBaseProcessDefinition[] GetProcDefs()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

An array of WFBaseProcessDefinition objects.

Example

```
IWFWorkflowService svc = GetWorkflowService();

try
 {
    //Returns Array of WFBaseProcessDefinition type.
    WFBaseProcessDefinition[] processDefinitions =
 svc.GetProcDefs();
    for (int i = 0; i < processDefinitions.Length; i++)
        {
        Console.WriteLine("Definition ID: '" +
        processDefinitions[i].DefID + "' ");
        Console.WriteLine("Definition Name: '" +
        processDefinitions[i].DefName + "' ");
        }
 }

catch (Exception ex)
 {
    Console.WriteLine(ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Get Process Definition Graphics

Description

Retrieves graphical data for the process definition in XML format. The graphical representation of the process is XML-serialized by the class Graphic Image. The graphical data is used to display the process visually.

Syntax

public virtual string GetProcDefGraphics(string pID)

Parameters

Name	Type	Description
PID	string	The process definition ID for a released process definition.

Output

Graphics object in XML format.

Example

```
// This is console application sample
IWFWorkflowService svc = GetWorkflowService();
//process definition ID or process instance ID
string procesID = … // for example,
"42544811EC2D4FC18E6BA15CC9FE28DF";

try
 {
    //returns an image of a process definition as string.
    string procDefGraphicsXML =
 svc.GetProcDefGraphics(procesID);
    GraphicImage g = new GraphicImage();
    g.FromXml(procDefGraphicsXML);
    byte[] images = g.Image // process image
    NamedRectangle[] shapes = g.Shapes;
 }

catch (Exception ex)
 {
```

```
    Console.WriteLine(ShUtil.GetSoapMessage(ex));
 }
/* This example produces the following results:
process definition Graphics:
<?xml version="1.0" encoding="utf-8"?><Graphics
left="3.33333333333333" right="5.76002857553708"
top="10.3848753378378" bottom="7"><Shapes><Shape
left="3.33333333333333" right="3.83333333333333"
top="10.3854166666667" bottom="9.88541666666667" name="Start"
/><Shape left="5.26041666666667"
right="5.76041666666667" top="7.5" bottom="7" name="Stop"
/><Shape left="4.30208333333333"
right="5.30208333333333" top="9.16666666666667"
bottom="8.66666666666667" name="Text File Writer.3"
/></Shapes><Image>@64R0lGODlh6QBFAXAAACwAAAAA6QB
FAYcAAAAMDAwkJCQsLCydAAClAACqAACuAACyAAC2AAC6AAC+
\nAADCAADGAADK
AAA8PDzOAADSAA
.................
\nTEGgS0+D2pKSHjUEWOzmRMPZljdstSBbQshYsxjRfu7xD
REAaV3vGgai7gAEaA3sj3bghhl2dbEN
\nDAM5bsCFff6yI1uYAAZ8WtowAasI
wudCT76Yxw1tYQa9/W1ww8ADCmxp+/SHxwvUm4UYBGq02d2S\nWJwAf
+brX/9H0ic
BCEbggcpuAAbuvO+YgBV8acwAwNO3PjK
+j4EKZzhNwp1PZ4dvfBUAoMEjgHG
b\nBGNA0rYGa1W+cpazvMA32cOlMo6cgAAAOw==</Image></Graphics>
*/
```

Supported Versions

3.2.0.4 and higher

Get Process Definition Name and Version

Description

Retrieves the process definition name and version.

Syntax

public virtual KeyValue GetProcDefNameVersion(string pID)

Parameters

Name	Type	Description
PID	string	The process definition ID for a released process definition.

Output

KeyValue object, where Key contains process definition name and Value contains version.

Example

```
// This is console application sample
IWFWorkflowService svc = GetWorkflowService();

//process definition ID for a process.
string processTemplatedID = … // for example
 "1e3d514d43d3465cae6ec3bbbd409168";

try
 {
    //Returns KeyValuepair, for example "process definition
    Name-process definition Version"
    KeyValue keyValue = _
 svc.GetProcDefNameVersion(processTemplatedID);
    Console.WriteLine("process definition Name: '" +
    keyValue.Key.Tostring() + "' ");
    Console.WriteLine("process definition Version: '" +
    keyValue.Value.Tostring() + "' ");
 }

catch (Exception ex)
 {
    Console.WriteLine( ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Get Process Definition XML

Description

Retrieves a process definition in XML format.

Syntax

public virtual string GetProcDefXml(string pID)

Parameters

Name	Type	Description
PID	string	The process definition ID for a released process definition.

Output

string that contains XML format of the process definition.

Example

```
// This is console application sample
IWFWorkflowService svc = GetWorkflowService();
//process definition ID for a process.
string processDefinitionID = ..// for example
 "42544811EC2D4FC18E6BA15CC9FE28DF";

try
 {
    //Returns process definition in XML format.
    string procDefXML =
 svc.GetProcDefXml(processDefinitionID);
    Console.WriteLine("process definition XML: \n");
    Console.WriteLine(procDefXML);
 }

catch (Exception ex)
 {
    Console.WriteLine( ShUtil.GetSoapMessage(ex));
 }

/*
```

```
This example produces the following results:
process definition XML:
<?xml version="1.0" encoding="utf-8"
standalone="no"?>\n<?wfmc-xpdl xmlns="http://
www.wfmc.org/2002/XPDL1.0"
xmlns:xpdl="http://www.wfmc.org/2002/XPDL1.0"
xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0"?>\n<!--
Process
Definition, Copyright 2003-2004 AgilePoint Inc., All
Rights Reserved.-->\n<ProcessDefinition
defName="TextFileWriterProcess" preVersion="" version="1.0"
description="" owner="Bipin.Shah" docRef=""
.................................
\nTEGgS0+D2pKSHjUEWOzmRMPZljdstSBbQshYsxjRfu7
xDREAaV3vGgai7gAEaA3sj3bghhl2dbEN\nDAM5bsCFff6
yI1uYAAZ8WtowAasIwudCT76Yxw1tYQa9/W1ww8ADCmxp+/
SHxwvUm4UYBGq02d2S\nWJwAf+brX/9H
0icBCEbggcpuAAbuvO+YgBV8acwAwNO3PjK+j4EKZzhNwp
1PZ4dvfBUAoMEjgHGb\nBGNA0rYGa1W+cpazvMA32cOlMo6cgAAAOw==</
Image>\n
</Graphics>\n</ProcessDefinition>
*/
```

Supported Versions

3.2.0.4 and higher

Get Released Process Definition ID

Description

Retrieves the released process definition ID by a specified process definition name.

Syntax

public virtual string GetReleasedPID(string pName)

Parameters

Name	Type	Description
processDefinitionName	string	The name of the process definition.

Output

The ID for the released process definition.

Example

```
//GetReleasedPID
IWFWorkflowService svc= GetWorkflowService();
string processDefinitionName = "Budget Request Approval
 Process";
string processDefinitionID =
svc.GetReleasedPID(processDefinitionName);
Console.WriteLine("Process definition ID=" +
 processDefinitionID);
```

Supported Versions

3.2.0.4 and higher

Get Released Process Definitions

Description

Retrieves the names and IDs of all released process definitions.

Syntax

public virtual KeyValue[] GetReleasedProcDefs()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

KeyValue array for pairs of process definition IDs and process definition names.

Example

```
KeyValue[] defs = svc.GetReleasedProcDefs();
for (int i = 0; i < defs.Length; i++)
    {
    Console.WriteLine("Key=" + defs[i].Key.Tostring() +
 "************"+"Value=" + defs[i].Value.Tostring());
    }
```

Supported Versions

3.2.0.4 and higher

Get Process Definition By Base Process Definition ID

Description

Retrieves all process definitions by a specified base process definition ID.

Syntax

public virtual WFBaseProcessDefinition[] GetProcDefByBasePID(string basePID)

Parameters

Name	Type	Description
basePID	string	The ID of the base process definition.

Output

Array of WFBaseProcessDefinition objects.

Example

```
// This is console application
IWFWorkflowService svc = GetWorkflowService();

//Base process definition ID.
```

```
string baseprocessInstanceID = … // for example
 "1e3d514d43d3465cae6ec3bbbd409168";

try
 {
    //Returns Array of WFBaseProcessDefinition for all
 versions of
    process definition WFBaseProcessDefinition[]
 processDefinitions =
    svc.GetProcDefByBasePID(baseprocessInstanceID);
    for (int i = 0; i < processDefinitions.Length; i++)
    {
       Console.WriteLine("Definition ID: '" +
 processDefinitions[i].DefID + "' ");
       Console.WriteLine("Definition Name: '" +
       processDefinitions[i].DefName + "' ");
    }
 }

catch (Exception ex)
 {
    Console.WriteLine(ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Update Process Definition

Description

Updates a process definition without using version control. This method is intended for minor changes only, such as typographical errors. Warning: Changes made using this method circumvent version control, meaning changes are not tracked, and versions cannot be managed. Do not use this call for making any major changes to the process definition.

Syntax

public virtual string UpdateProcDef(string xml)

Parameters

Name	Type	Description
xml	string	A string that contains process definition in XML format. To generate the process definition file in XML format, in AgilePoint Envision, click **File > Export & Import > Save As Deploying File(xml)**. You can also download the process definition XML from AgilePoint Enterprise Manager.

Output

Returns the unique identifier for the process definition that is updated.

Example

```
// This is console application sample
IWFWorkflowService svc = GetWorkflowService();
string processDefinitionXML =..// see previous description of
 how to
get process definition XML

try
 {
    //Update Process definition using updated process xml
 string
    string processDefinitionID = svc.UpdateProcDef(xml);
 }

catch (Exception ex)
 {
    Console.WriteLine("Exception:" +
 ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

4.0.1 and higher

Methods for Process Instances

This section describes service calls related to process instances.

Create Process Instance

Description

Creates a process instance for a specified process definition ID, and parameters.

Syntax

public virtual WFEvent CreateProcInst(string PID, string PIID, string PIName, string workObjectID, string superPIID, bool startImmediately)

Parameters

Name	Type	Description
PID	string	The process definition ID for a released process definition.
PIID	string	A 32 character unique process instance ID for the process instance you are creating. If you set this value to null, the AgilePoint Server generates the ID.
PIName	string	A unique process name that is associated with the process definition. The maximum length of process instance name is 1024 characters.
workObjectID	string	A 256-character ID for an object, such as a document, that is associated with the process instance. (Even though the field size is 256 characters, in common practice, this will usually return a 32-character GUID.)

Name	Type	Description
superPIID	string	A 32-character unique process instance ID that acts as a parent process instance of the process instance that is intended to create. In other words, this is the ID of the process instance on which you want to base your new process instance.
startImmediately	bool	An obsolete, legacy parameter that must be true.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string processDefinitionName = "EmployeeOnboardProcess";

// get UUID of released process definition
string processDefinitionID =
 svc.GetReleasedPID(processDefinitionName);

// assign UUID of process instance
string processInstanceID = UUID.GetID();

// process instance name that has to be unique within process
 definition ID
string processInstanceName = string.Format("{0}-{1}",
processDefinitionName DateTime.Now.Ticks );

// work object ID
string workObjectID = UUID.GetID();

// create process instance
WFEvent event = svc.CreateProcInst( pID, piID, piName,
 workObjectID, null, true);
```

Supported Versions

3.2.0.4 and higher

Create Process Instance (Extended with Initiator and Work Object Info)

Description

Creates a process instance with a specified workObjectInfo value. The workObjectInfo parameter provides additional information about a work object, such as a URL for a document.

Syntax

public virtual WFEvent CreateProcInstEx(string PID, string PIID, string PIName, string workObjectID, string workObjectInfo, string superPIID, string initiator, string customID, NameValue[] attributes, bool startImmediately)

Parameters

Name	Type	Description
PID	string	The process definition ID for a released process definition.
PIID	string	A 32 character unique process instance ID for the process instance you are creating. If you set this value to null, the AgilePoint Server generates the ID.
PIName	string	A unique process name that is associated with the process definition. The maximum length of process instance name is 1024 characters.
workObjectID	string	A 256-character ID for an object, such as a document, that is associated with the process instance. (Even though the field size is 256 characters, in common practice, this will usually return a 32-character GUID.)
workObjectInfo	string	A 1024-character string associated with the process instance. Usually this parameter

Name	Type	Description
		is used to hold more information about the work object, such as a URL for a document, within the process instance.
superPIID	string	A 32-character unique process instance ID that acts as a parent process instance of the process instance that is intended to create. In other words, this is the ID of the process instance on which you want to base your new process instance.
initiator	string	A string that contains the user name of the user who initiates process.
customID	string	A work object ID as identified in the process instance. This method allows only one custom ID (work object ID).
attributes	NameValue	Name-value pairs associated with a custom ID.
startImmediately	bool	An obsolete, legacy parameter that must be true.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
… see previous sample
string workObjectInfo = ..// for example, a XML-serialized of
 an object
WFEvent evt = svc.CreateProcInstEx(
                  processDefinitionID,
                  processInstanceID,
                  processInstanceName,
                  workObjectID,
                  workObjectInfo,
                  parentProcessInstanceID,
                  initiator,
                  workObjectID,
                  ds.ToArray(),
```

```
                    true);
```

Supported Versions

4.0.1 and higher

Create Process Instance (Extended with Initiator)

Description

Creates a process instance in which the user name for the user who initiates the process is specified.

Syntax

public virtual WFEvent CreateProcInstEx(string PID, string PIID, string PIName, string workObjectID, string superPIID, string initiator, string customID, NameValue[] attributes, bool startImmediately)

Parameters

Name	Type	Description
PID	string	The process definition ID for a released process definition.
PIID	string	A 32 character unique process instance ID for the process instance you are creating. If you set this value to null, the AgilePoint Server generates the ID.
PIName	string	A unique process name that is associated with the process definition. The maximum length of process instance name is 1024 characters.
workObjectID	string	A 256-character ID for an object, such as a document, that is associated with the process instance. (Even though the field size is 256 characters, in common practice, this will usually return a 32-character GUID.)

Name	Type	Description
superPIID	string	A 32-character unique process instance ID that acts as a parent process instance of the process instance that is intended to create. In other words, this is the ID of the process instance on which you want to base your new process instance.
initiator	string	A string that contains the user name of the user who initiates process.
customID	string	A work object ID as identified in the process instance. This method allows only one custom ID (work object ID).
attributes	NameValue	Name-value pairs associated with a custom ID.
startImmediately	bool	An obsolete, legacy parameter that must be true.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
… see previous sample
string initiator = ..// System.Environment.UserName
WFEvent evt = svc.CreateProcInstEx(
                  processDefinitionID,
                  processInstanceID,
                  processInstanceName,
                  workObjectID,
                  parentProcessInstanceID,
                  initiator,
                  workObjectID,
                  ds.ToArray(),
                  true);
```

Supported Versions

3.2.0.4 and higher

Create Process Instance (Extended Method)

Description

Creates a process instance that can have additional input arguments added to the function.

Syntax

public virtual WFEvent CreateProcInstEx(string PID, string PIID, string PIName, string workObjectID, string superPIID, string customID, NameValue[] attributes, bool startImmediately)

Parameters

Name	Type	Description
PID	string	The process definition ID for a released process definition.
PIID	string	A 32 character unique process instance ID for the process instance you are creating. If you set this value to null, the AgilePoint Server generates the ID.
PIName	string	A unique process name that is associated with the process definition. The maximum length of process instance name is 1024 characters.
workObjectID	string	A 256-character ID for an object, such as a document, that is associated with the process instance. (Even though the field size is 256 characters, in common practice, this will usually return a 32-character GUID.)
superPIID	string	A 32-character unique process instance ID that acts as a parent

Name	Type	Description
		process instance of the process instance that is intended to create. In other words, this is the ID of the process instance on which you want to base your new process instance.
customID	string	A work object ID as identified in the process instance. This method allows only one custom ID (work object ID).
attributes	NameValue	Name-value pairs associated with a custom ID.
startImmediately	bool	An obsolete, legacy parameter that must be true.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string processDefinitionName = "EmployeeOnboardProcess";

// get UUID of released process definition
string processDefinitionID =
svc.GetReleasedPID(processDefinitionName);

// assign UUID of process instance
string processInstanceID = UUID.GetID();

// process instance name that has to be unique within process
 definition ID
string processInstanceName = string.Format("{0}-{1}",
processDefinitionName DateTime.Now.Ticks );

// work object ID
string workObjectID = UUID.GetID();
//parent process instance ID is required if this is to create
 a sub
process. If not, just provide null
string parentProcessInstanceID = .. // for example,
"09315f0ae769429bbfb243f888bcb09f" or null
List<NameValue> ds = new List<NameValue>();
ds.Add(new NameValue("CustomAttrKey1", "CustomAttrValue1"));
```

```
ds.Add(new NameValue("CustomAttrKey2", true));
ds.Add(new NameValue("CustomAttrKey3", 12345));
WFEvent evt = svc.CreateProcInstEx(
                  processDefinitionID,
                  processInstanceID,
                  processInstanceName,
                  workObjectID,
                  parentProcessInstanceID,
                  workObjectID,
                  ds.ToArray(),
                  true);
```

Supported Versions

3.2.0.4 and higher

Cancel Process Instance

Description

Cancels the process instance based on a specified process instance identifier. This method cancels all automatic work items, manual work items, and child process instances.

Syntax

public virtual WFEvent CancelProcInst(string piID)

Parameters

Name	Type	Description
processInstanceID	string	A string that contains the ID of the process instance.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = GetWorkflowService();
```

```
string processInstanceID = ..// the ID of the process instance
 to be
canceled.

try
 {
    WFEvent evt = svc.CancelProcInst(processInstanceID);
 }

catch( Exception ex)
 {
    base.ShowMessage( base.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Delete Process Instance

Description

Deletes a process instance. This method removes the specified process instance and all of associated data from database, such as work items, email, and activity instances associated with this process instance. It may take some time to complete this transaction.

Syntax

public void DeleteProcInst(string piID)

Parameters

Name	Type	Description
processInstanceID	string	A string that contains the ID of the process instance.

Output

None.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string processInstanceID = ..// the ID of the process instance
 to be
canceled.

try
 {
    WFEvent evt = svc.DeleteProcInst(processInstanceID);
 }

catch( Exception ex)
 {
    base.ShowMessage( base.GetSoapMessage(ex));
 }
```

Supported Versions

4.0.1 and higher

Merge Process Instance

Description

Merges 2 or more process instances into one process instance.

These process instances should be based on the same process definition.

Syntax

public virtual string MergeProcInsts(WFProcessMergingInstruction instruction)

Parameters

Name	Type	Description
instruction	WFProcessMergingInstruction	An object that specifies the instructions for the merge.

Output

The process instance ID of the merged process instance.

Example

```
string MergeProcessInstances(IWFWorkflowService svc, string[]
processInstanceIDs)
 {
    IWFWorkflowService svc = GetWorkflowService();

    // suspends all of process instances to be merged.
    foreach (string id in processInstanceIDs)
        {
        svc.SuspendProcInst(id);
        }

    // query process instances
    string inExpr = ShUtil.Merge(processInstanceIDs, true);
    WFQueryExpr queryExpr = new WFQueryExpr("PROC_INST_ID",
 SQLExpr.IN,
    WFAny.Create(inExpr), true);
    WFBaseProcessInstance[] pis =
 svc.QueryProcInsts(queryExpr);

    // merge custom attributes
    NameValue[] mergedCustomAttributes =
 GetMergeCustomAttributes(api, pis);
    string procInstID = UUID.GetID();
    string procInstName = pis[0].DefName + "_" +
 DateTime.Now.Tostring()
    + " - Merged";
    string workObjectID = procInstID + " - Merged";

    WFProcessMergingInstruction instruction = new
    WFProcessMergingInstruction();
    instruction.MergingProcessInstanceIDs =
 processInstanceIDs;
    instruction.MergedProcessInstance = new
    WFProcessMergingInstruction.MergedProcessParameter(
                  procInstID,
                  procInstName,
                  workObjectID,
                  null,
                  mergedCustomAttributes);
    instruction.Validate();
    return svc.MergeProcInsts(instruction);
 }

// sample code for merging custom attributes
```

```
private NameValue[]
 GetMergeCustomAttributes(IWFWorkflowService svc,
WFBaseProcessInstance[] pis)
 {
    List<string> workObjectIDs = new List<string>();
    foreach (WFBaseProcessInstance pi in pis)
        {
        workObjectIDs.Add(pi.WorkObjectID);
        }
    KeyValue[] items =
 svc.GetCustomAttrsEx(workObjectIDs.ToArray());
    Dictionary<string, System.Xml.XmlDocument> dss =
    new Dictionary<string, System.Xml.XmlDocument>();
    foreach (KeyValue item in items)
        {
        WFCustomAttributes ds = new WFCustomAttributes();
        ds.AttrXml = item.Value;
        System.Xml.XmlDocument xmlDoc = new
 System.Xml.XmlDocument();
        xmlDoc.LoadXml(ds["//"] as string);
        dss[item.Key] = xmlDoc;
        }
    // master processInstanceID and workObjectID
    string masterProcessInstanceID = pis[0].ProcInstID;
    string masterWorkObjectID = pis[0].WorkObjectID;
    System.Xml.XmlDocument masterXmlDoc =
 dss[masterWorkObjectID];
    System.Xml.XmlNamespaceManager nsm =
    ShUtil.GetNamespaces(masterXmlDoc);
    System.Xml.XmlNode titleNode =
    masterXmlDoc.SelectSingleNode("/pd:issueTracking/
pd:issueTitle", nsm);
    titleNode.InnerText = "Title - Merged";
    System.Xml.XmlNode descriptionNode =

    masterXmlDoc.SelectSingleNode("/pd:issueTracking/
pd:description", nsm);
    descriptionNode.InnerText = "Description - Merged";
    System.Xml.XmlNode parent =
    masterXmlDoc.SelectSingleNode("/pd:issueTracking/
pd:Persons", nsm);

    // merge rest of children
    foreach (string key in dss.Keys)
        {
        if (key == masterWorkObjectID) continue;
        System.Xml.XmlDocument secondaryXmlDoc = dss[key];
        System.Xml.XmlNamespaceManager nsmgr =
        ShUtil.GetNamespaces(secondaryXmlDoc);
        System.Xml.XmlNode node =
        secondaryXmlDoc.SelectSingleNode("/pd:issueTracking/
pd:Persons", nsmgr);
        parent.InnerXml += node.InnerXml;
        }
    return NameValue.Array("//", masterXmlDoc.OuterXml);
```

```
 }

#endregion
```

Supported Versions

4.5 and higher

Migrate Process Instances

Description

Migrates a process definition from one version to another version.

Syntax

public virtual void MigrateProcInst(WFProcessMigrationInstruction instruction, string processInstanceID, string reserved)

Parameters

Name	Type	Description
instruction	WFProcessMigrationInstruction	An object that specifies the instructions for the migration.
processInstanceID	string	A string that contains the ID of the process instance.
reserved	string	null.

Output

```
IWFWorkflowService svc = GetWorkflowService();
WFProcessMigrationInstruction pmi = new
WFProcessMigrationInstruction();

// some code...
svc.MigrateProcInst(pmi, currentProcessInstanceID, null);

// some more code...
```

Example

```
IWFWorkflowService svc = GetWorkflow();
string processInstanceID = "DB50CFEFDE464A78AAAA9BD7D6E6D9D0";
WFProcessMigrationInstruction pmi = new
WFProcessMigrationInstruction();

//add the correct CurrentActivityUniqueName
string currentActivityUniqueName = "BudgetRequest";

//add the correct TargetActivityUniqueName
string targetActivityUniqueName = "BudgetRequestNew";
bool bCurrentActivated = false;
pmi.AddMatchingActivity(currentActivityUniqueName,targetActivityUniqueName,
bCurrentActivated);
svc.MigrateProcInst(pmi, currentProcessInstanceID, null);
```

Supported Versions

4.0.1 and higher

Promote Process Instance

Description

Promotes a process instance. This method is obsolete.

Syntax

public virtual WFEvent PromoteProcInst(String piID)

Parameters

Name	Type	Description
PIID	string	A 32 character unique process instance ID for the process instance you are creating. If you set this value to null, the AgilePoint Server generates the ID.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
None.
```

Supported Versions

3.2.0.4 and higher

Rollback Process Instance

Description

Rolls a process instance back to a previous specified activity, or skips a specified activity if has not yet been completed. When this method is invoked, the current or skipped activity becomes canceled. When skipping, the process moves forward regardless of the activity's status.

Syntax

public virtual WFEvent RollbackProcInst(String AIID)

Parameters

Name	Type	Description
activityInstanceID	string	A string that contains the ID for an activity instance.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
//This is console application sample
IWFWorkflowService svc = GetWorkflowService();
string activityInstanceID = ..// target activity instance to
 roll back

try
 {
    WFEvent evt =
 workflowService.RollbackProcInst(activityInstanceID);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Split Process Instance

Description

Splits one process instance into 2 or more process instances. The original process is canceled.

Syntax

public virtual string[] SplitProcInst(WFProcessSplittingInstruction instruction)

Parameters

Name	Type	Description
instruction	WFProcessSplitting	An object that specifies the instructions for splitting the process instance.

Output

A collection of strings that contain the process instance IDs for the process instances that were created from the split.

Example

```
// This is console application sample
public string[] SplitProcessInstance(string processInstanceID)
 {
    IWFWorkflowService svc = GetWorkflowService();
    WFBaseProcessInstance processInstance =
    svc.GetProcInst(processInstanceID);
    WFCustomAttributes ds = ds = new
    WFCustomAttributes(processInstance.WorkObjectID);
    ds.AttrXml =
 svc.GetCustomAttrs(processInstance.WorkObjectID);
    string xml = ds["//"] as string;
    Console.WriteLine("Splitting custom attributes ...");
    List<NameValue[]> splittedCustomAttributes =
    GetSplittedCustomAttributes(xml);
    WFProcessSplittingInstruction instruction;
    instruction = new WFProcessSplittingInstruction();
    instruction.SplittingProcessInstanceID = procInstID;
    List<string> workObjectIDs = new List<string>();
    for (int i = 0; i < splittedCustomAttributes.Count; ++i)
        {
        string splittedProcInstID = UUID.GetID();
        string splittedProcInstName = string.Format("{0} –
 Splitted -
     {1}", pi.ProcInstName, i + 1);
        string splittedWorkObjectID = string.Format("{0} –
 Splitted -
     {1}", pi.WorkObjectID, i + 1);
        instruction.Add(
        splittedProcInstID,
        splittedProcInstName,
        splittedWorkObjectID,
        null,
        splittedCustomAttributes[i]);
        }
    instruction.Validate();
    Console.WriteLine("Suspending process instance...");
    svc.SuspendProcInst(procInstID);
    System.Threading.Thread.Sleep(1000);
    string[] ids = svc.SplitProcInst(instruction);
    return ids;
 }

// function to split custom attributes
```

```
private List<NameValue[]> GetSplittedCustomAttributes(string
 xml)
 {
    System.Xml.XmlDocument xmlDoc = new
 System.Xml.XmlDocument();
    xmlDoc.PreserveWhitespace = true; xmlDoc.LoadXml(xml);
    System.Xml.XmlNamespaceManager nsm =
 ShUtil.GetNamespaces(xmlDoc);
    System.Xml.XmlNode parent =
    xmlDoc.SelectSingleNode("/pd:issueTracking/pd:Persons",
 nsm);

    // split repeating notes – 'pd:issueTracking/pd:Persons/
pd:Person'
    System.Xml.XmlNodeList nodes =
 parent.SelectNodes("pd:Person", nsm);

    if (nodes == null || nodes.Count < 2)
    {
       throw new InvalidOperationException("Failed to get
 splitted
       customAttributes,
             'pd:issueTracking/pd:Persons/pd:Person' does not
 have
       multiple node.");
    }

    // remove child nodes
    System.Xml.XmlNode titleNode =
    xmlDoc.SelectSingleNode("/pd:issueTracking/pd:issueTitle",
 nsm);
    string title = titleNode.InnerText;
    System.Xml.XmlNode descriptionNode =
    xmlDoc.SelectSingleNode("/pd:issueTracking/
pd:description", nsm);
    string description = descriptionNode.InnerText;
    List<NameValue[]> splittedCustomAttributes = new
    List<NameValue[]>();
    for( int i = 0; i < nodes.Count; ++i)
    {
       //change title
       titleNode.InnerText = title + "-Splitted-" + i;
       descriptionNode.InnerText = description + "- Splitted-"
 + i;
       parent.RemoveAll();
       parent.AppendChild(nodes[i]);
       splittedCustomAttributes.Add( NameValue.Array("//",
       xmlDoc.OuterXml) );
    }
    return splittedCustomAttributes;
 }
```

Supported Versions

4.5 and higher

Start Process Instance

Description

Starts a process instance that is not set to start immediately on creation. This method is obsolete.

Syntax

public virtual WFEvent StartProcInst(string piID)

Parameters

Name	Type	Description
processInstanceID	string	A string that contains the ID of the process instance.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
None.
```

Supported Versions

3.2.0.4 and higher

Suspend Process Instance

Description

Suspends a process instance. The process instance status is changed to Suspended, and the statuses of all the work items (tasks) become Pending.

Syntax

public virtual WFEvent SuspendProcInst(string piID)

Parameters

Name	Type	Description
processInstanceID	string	A string that contains the ID of the process instance.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
// This is console application sample
IWFWorkflowService svc = GetWorkflowService();

try
 {
    string processInstanceID = …// process instance to be
 suspended.
    WFEvent event = svc.SuspendProcInst(processInstanceID);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed: " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Get Events By Process Instance ID

Description

Retrieves all the events that have occurred for a specified process instance.

Syntax

public virtual WFEvent[] GetEventsByProcInstID(string piID)

Parameters

Name	Type	Description
processInstanceID	string	A string that contains the ID of the process instance.

Output

An array ofWFEvent objects.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string processInstanceID = … // process instance ID

try
 {
    WFEvent[] events =
 svc.GetEventsByProcInstID(processInstanceID);
    for (int i = 0; i < events.Length; i++)
        {
        Console.WriteLine("Event ID: '" + events[i].EventID +
 "' ");
        Console.WriteLine("Event Name: '" +
 events[i].EventName + "'");
        }
 }
```

```
catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Get Process Instance

Description

Retrieves basic information about a specified process instance.

Syntax

public virtual WFBaseProcessInstance GetProcInst(string PIID)

Parameters

Name	Type	Description
processInstanceID	string	A string that contains the ID of the process instance.

Output

WFBaseProcessInstance object that contains basic information about a process instance. It returns null if the process instance ID does not exist.

Example

```
// This is sample code for console application
IWFWorkflowService svc = GetWorkflowService();
string processInstanceID = …// process instance ID

try
 {
    //Returns an instance of WFBaseProcessInstance type.
    WFBaseProcessInstance processInstance =
    svc.GetProcInst(processInstanceID);
 }
```

```
catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Get Process Instance Attributes

Description

Retrieves multiple attributes of a process instance.

Syntax

public virtual NameValue[] GetProcInstAttrs(String piID)

Parameters

Name	Type	Description
PIID	string	A 32 character unique process instance ID for the process instance you are creating. If you set this value to null, the AgilePoint Server generates the ID.

Attributes

Name	Description
DefID	The ID of the process definition.
DefName	The name of the process definition.
ProcInstName	The name of the process instance.
Status	The current status of the process instance.
SuperProcInstID	The parent process instance ID.

Name	Description
workObjectID	The ID of the work object.
StartDate	The date and time when the process instance was started.
DueDate	The date that the process instance is expected to be complete
LastModifiedDate	The date and time that the last modification was made to the process instance.

Output

Array of NameValue objects that holds the values of all the requested attributes.

Example

```
// This is console application sample
IWFWorkflowService svc = GetWorkflowService();
string processInstanceID = …// process instance ID
NameValue[] attributes =
 svc.GetProcInstAttrs(processInstanceID);
for (int i = 0; i < attributes.Length; i++)
 {
    NameValue nv = attributes[i];
    Console.WriteLine("Process Instance Attribute, {0}={1}",
 nv.Name, nv.Value);
 }
```

Supported Versions

3.2.0.4 and higher

Query Process Instances

Description

Retrieves a list of process instances that match a specified query expression. The WFQueryExpr string is used to generate a query expression, and the client application specifies the query terms.

Syntax

public virtual WFBaseProcessInstance[] QueryProcInsts(WFQueryExpr expr)

Parameters

Name	Type	Description
expr	WFQueryExpr	An object that represents the where clause of a SQL query expression.

Output

An array of WFBaseProcessInstance objects. It returns null if nothing matches to the query expression.

Example

```
IWFWorkflowService svc = GetWorkflowService();

// query all running process instance
string status = WFProcessInstance.RUNNING;
WFAny any = WFAny.Create(status);
WFQueryExpr expr = new WFQueryExpr("STATUS", SQLExpr.EQ, any,
 true);

try
 {
    // Calling the QueryProcInsts WebMethod, passing the
 expression as the argument.
    WFBaseProcessInstance[] result = svc.QueryProcInsts(expr);

    if (result != null)
        {
        // Iterating through the list of the Process Instance
        foreach (WFBaseProcessInstance processInstance in
 result
            {
            //Displaying the Process Instance Details on
 Console.
            Console.WriteLine("ApplName-->" +
            processInstance.ApplName);
            Console.WriteLine("DefName-->" +
 processInstance.DefName);
            Console.WriteLine("DefID-->" +
 processInstance.DefID);
            Console.WriteLine("CompletedDate-->" +
            processInstance.CompletedDate);
            Console.WriteLine("LastModifiedBy-->" +
            processInstance.LastModifiedBy);
            }
```

```
        }
 }

catch(Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Query Process Instances (Extended Method)

Description

Retrieves a list of process instances with a SQL query expression specified by the client application.

Syntax

public virtual WFBaseProcessInstance[] QueryProcInstsEx(string sql)

Parameters

Name	Type	Description
sql	string	A string that contains the where clause of the SQL statement you want to query.

Output

An array ofWFBaseProcessInstance objects. It returns null if nothing matches the SQL query expression.

Example

```
// Console application sample code to illustrate
 QueryProcInstsEx API.
IWFWorkflowService svc = GetWorkflowService();

// SQL Expression
string where = "STATUS in ('Running','Canceled')";
```

```
try
 {
    //Calling QueryProcInstsEx WebMethod with sql query
 expression as argument.
    WFBaseProcessInstance[] result =
 svc.QueryProcInstsEx(where);

    if (result != null)
        {
        // Iterating through the list of the Process Instance
        foreach (WFBaseProcessInstance processInstance in
 result
            {
            //Displaying the Process Instance Details on
 Console.
            Console.WriteLine("ApplName-->" +
 processInstance.ApplName
            Console.WriteLine("DefName-->" +
 processInstance.DefName);
            Console.WriteLine("DefID-->" +
 processInstance.DefID);
            Console.WriteLine("CompletedDate-->" +
            processInstance.CompletedDate);
            Console.WriteLine("LastModifiedBy-->" +
            processInstance.LastModifiedBy);
            }
        }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Update Process Instance

Description

Updates attributes of a workflow process instance. The attributes that can be updated are listed in the attribute table.

Syntax

public void UpdateProcInst(string processInstanceID, NameValue[] attributes)

Parameters

Name	Type	Description
processInstanceID	string	A string that contains the ID of the process instance.
attributes	NameValue	Name-value pairs associated with a custom ID.

Attributes

Name	Description
ProcInstName	The name of the process instance.
DueDate	The date that the process instance is expected to be complete
workObjectID	The ID of the work object.

Output

None.

Example

```
// This is console application sample to update process
 instance name
IWFWorkflowService svc = GetWorkflowService();
string processInstanceID = … // process instance ID
string newProccessInstanceName = "[new process instance
 name]";
DateTime newDueDate = DateTime.Now.AddDays(7.0);

try
 {
    WFBaseProcessInstance inst =
 svc.GetProcInst(processInstanceID);
    NameValue[] processInstanceAttributes = new NameValue[]
        {
       new NameValue("ProcInstName", newProccessInstanceName),
       new NameValue("DueDate", newDueDate)),
        };

    // update process instance
```

```
    svc.UpdateProcInst(processInstanceID,
 processInstanceAttributes);

    // check if it has been updated.
    string processInstanceName =
    svc.GetProcInst(processInstanceID).ProcInstName;
    Console.WriteLine("New Process Instance Name= '{0}",
    processInstanceName);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Methods for Activity Instances

This section describes service calls related to activity instances.

Cancel Activity Instance

Description

Cancels a manual activity instance. along with all manual work items associated with the specified manual activity instance ID. Note that an activity instance can be associated with one or more manual work items. One the manual activity instance is canceled, the process instance will move forward to the next activity.

Syntax

public virtual WFEvent CancelActivityInst(string activityInstanceID)

Parameters

Name	Type	Description
activityInstanceID	string	A string that contains the ID for an activity instance.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string activityInstanceInstanceID = …// activity instance
 needs to be
canceled.

try
    {
    WFEvent evt =
 svc.CancelActivityInst(activityInstanceInstanceID);
    }

catch(Exception ex)
    {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Get Activity Instance

Description

Retrieves basic information for a specified activity instance.

Syntax

public virtual WFBaseActivityInstance GetActivityInst(string aiID)

Parameters

Name	Type	Description
activityInstanceID	string	A string that contains the ID for an activity instance.

Output

WFBaseActivityInstance object.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string activityInstanceID = ..// activity instance

try
 {
    WFBaseActivityInstance activityInstance =
    _svc.GetActivityInst(activityInstanceID);
    Console.WriteLine("DisplayName" +
 activityInstance.DisplayName);
    Console.WriteLine("ID" + activityInstance.ID);
    Console.WriteLine("CompletedDate" +
 activityInstance.CompletedDate);
    Console.WriteLine("DueDate" + activityInstance.DueDate);
 }

catch( Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Get Activity Instance Status

Description

Retrieves all the status of all activity instances for a specified process instance.

Syntax

public virtual KeyValue[] GetActivityInstStatus(string piID)

Parameters

Name	Type	Description
processInstanceID	string	A string that contains the ID of the process instance.

Output

An array of KeyValue objects that holds pairs of activity definition names and statuses. The statuses can be Passed, Active, Pending, Activated, Canceled, or null.

Example

```
// This is console application sample.
IWFWorkflowService svc = GetWorkflowService();
string processInstanceID = ..// for example,
 "02C3FA88ADE04750A34B5B3168C25793";

try
 {
    KeyValue[] resultList =
 svc.GetActivityInstStatus(processInstanceID);
    foreach (KeyValue result in resultList)
    {
       System.Console.WriteLine("Activity Definition ID:
 '{0}'",result.Key);
       System.Console.WriteLine("Status: '{0}'",
       result.Value);
    }
 }

catch( Exception ex )
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Get Activity Instances By Process Instance ID

Description

Retrieves the status of all activity instances for a specified process instance.

Syntax

public virtual WFBaseActivityInstance[] GetActivityInstsByPIID(string piID)

Parameters

Name	Type	Description
processInstanceID	string	A string that contains the ID of the process instance.

Output

An array of WFBaseActivityInstance objects.

Example

```
// This is console application sample
IWFWorkflowService svc = GetWorkflowService();
string processInstanceID
 = ..//"02C3FA88ADE04750A34B5B3168C25793";

try
 {
    WFBaseActivityInstance[] activityInstance =
    svc.GetActivityInstsByPIID(processInstanceID);
    foreach (WFBaseActivityInstance activity in
 activityInstance)
        {
        System.Console.WriteLine("Activity DispalyName:
 '{0}'", activity.DisplayName);
        System.Console.WriteLine("CompletedDate: '{0}'",
 activity.CompletedDate);
     }
 }

catch( Exception ex)
 {
```

```
   Console.WriteLine("Failed! " +
ShUtil.GetSoapMessage(ex) );
   }
```

Supported Versions

3.2.0.4 and higher

Query Activity Instances

Description

Retrieves activity instances that match a query expression.

Syntax

public virtual WFBaseActivityInstance[] QueryActivityInsts(WFQueryExpr expr)

Parameters

Name	Type	Description
expr	WFQueryExpr	An object that represents the where clause of a SQL query expression.

Output

An array of WFBaseActivityInstance objects.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string processInstanceID = …
WFAny any = WFAny.Create(processInstanceID);
WFQueryExpr expr = new WFQueryExpr("PROC_INST_ID", SQLExpr.IN,
 any,
true);

try
 {
   WFBaseActivityInstance[] ais =
 svc.QueryActivityInsts(expr);
```

```
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Rollback Activity Instance

Description

Rolls back a manual activity instance to the token position EN – that is, the state where the activity is entered. All work items associated with the manual activity instance with the status of NEW, OVERDUE, or ASSIGNED are canceled.

Syntax

public virtual WFEvent RollbackActivityInst(string activityInstanceID)

Parameters

Name	Type	Description
activityInstanceID	string	A string that contains the ID for an activity instance.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
// This is console application sample
IWFWorkflowService svc = GetWorkflowService();
string activityInstanceID = …

try
  {
```

```
  //Rollbacking the activity instance
  WFEvent evt = svc.RollbackActivityInst(activityInstanceID);
  }

catch (Exception ex)
  {
  Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
  }
```

Supported Versions

3.2.0.4 and higher

Rollback Activity Instances

Description

Rolls back a process instance according to a specified instruction. The class WFPartialRollbackInstructionis used to specify detailed information about the instruction.

Syntax

public virtual WFEvent RollbackActivityInsts(WFPartialRollbackInstruction instruction)

Parameters

Name	Type	Description
instruction	WFPartialRollbackInstruction	An object that specifies the instructions for the partial rollback.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
//Sample for partial rollback
IWFWorkflowService svc = GetWorkflowService();

// PartialRollback unit
```

```
WFPartialRollbackInstruction.PartialRollbackUnit unit1
 = new WFPartialRollbackInstruction.PartialRollbackUnit();
unit1.DestinationActivityInstanceID = ... // destination
 activity
instance ID
unit1.SourceActivityInstanceIDs = new string[] { ... }; //
 array of
source activity instance ID
WFPartialRollbackInstruction.PartialRollbackUnit unit2
 = new WFPartialRollbackInstruction.PartialRollbackUnit();
unit2.DestinationActivityInstanceID = ... // destination
 activity
instance ID
unit2.SourceActivityInstanceIDs = new string[] { … }; // array
 of
source activity instance ID
WFPartialRollbackInstruction instruction = new
WFPartialRollbackInstruction();

try
 {
    //Rollbacking the activity instance
    WFEvent evt = workflowService.
 RollbackActivityInsts(instruction);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

4.6 and higher

Methods for Automatic Work Items (Procedures)

This section describes service calls related to automatic work items.

Cancel Procedure

Description

Cancels an automatic work item based on supplied specified automatic work item identifier.

Syntax

public virtual WFEvent CancelProcedure(string wID)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string workItemID = …//

try
 {
    WFEvent evt = svc.CancelProcedure(workItemID);
 }

catch( Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Complete Procedure

Description

Marks an automatic work item as completed by an asynchronous activity.

Syntax

public virtual WFEvent CompleteProcedure(string wID)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string autoWorkItemID = ..//

try
 {
    svc.CompleteProcedure(autoWorkItemID);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Get Procedure

Description

Retrieves work item data by a specified work item ID.

Syntax

public virtual WFAutomaticWorkItem GetProcedure(string wID)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.

Output

WFAutomaticWorkItem object.

Example

```
IWFWorkflowService svc = GetWorkflowService()
string workItemID = ..// for example,
"54A648A0A3004A02981E7F0848820FE7";

try
 {
    WFAutomaticWorkItem wItem = svc.GetProcedure(workItemID);
 }

catch( Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Query Procedure List

Description

Retrieves a list of automatic work items that match a specified query expression.

Syntax

public virtual WFAutomaticWorkItem[] QueryProcedureList(WFQueryExpr expr)

Parameters

Name	Type	Description
expr	WFQueryExpr	An object that represents the where clause of a SQL query expression.

Output

An array of automatic work items.

Example

```
IWFWorkflowService svc = GetWorkflowService();

try
 {
    //WebMethod with sql query expression as argument.
    WFAny any = WFAny.Create(WFAutomaticWorkItem.WAITING);
    WFQueryExpr expr = new WFQueryExpr("STATUS", SQLExpr.EQ,
 any, true);
    WFAutomaticWorkItem[] result =
 svc.QueryProcedureList(expr);

    if (result != null)
        {
        // Iterating through the list of the automatic work
 item
        foreach (WFAutomaticWorkItem re in result)
            {
            Console.WriteLine("ActivityInstID-->" +
                re.ActivityInstID);
            Console.WriteLine("ApplName-->" + re.ApplName);
            Console.WriteLine("ProcInstID-->" +
 re.ProcInstID);
            Console.WriteLine("CreatedDate-->" +
 re.CreatedDate);
            }
        }
 }

catch (Exception ex)
```

```
{
   Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
}
```

Supported Versions

3.2.0.4 and higher

Methods for Manual Work Items (Tasks)

This section describes service calls related to manual work items (tasks).

Assign Work Item

Description

Assigns a work item to a user, which often means claiming a work item for oneself. This is often used with task pools where work items are created, and then multiple users are notified, but the work item is not immediately assigned to a user. A user then claims the work item, or his manager assigns it to him. The user must have privileges to claim or assign the work item.

Syntax

public virtual WFEvent AssignWorkItem(string wID)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.

Output

WFEvent class that represents the workflow event instance that is created when the task is assigned.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string workItemID = …// work item ID
```

```
try
    {
    WFEvent evt = svc.AssignWorkItem(workItemID);
    }

catch( Exception ex)
    {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Assign Work Item (Extended Method)

Description

Assigns a work item to a user, which often means claiming a work item for oneself. This is often used with task pools where work items are created, and then multiple users are notified, but the work item is not immediately assigned to a user. A user then claims the work item, or his manager assigns it to him. The user must have privileges to claim or assign the work item. This method extends AssignWorkItem() by allowing you to specify client data.

Syntax

public virtual WFEvent AssignWorkItemEx(string wID, string clientData)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.
clientData	string	A string that contains the client data. If clientData is null, the system will keep existing client data, otherwise the relevant data would be overwritten.

Output

WFEvent object representing the workflow event instance raised by the invocation of the work assignment.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string wID = …// work item ID
string clientData = null;

try
    {
    WFEvent evt = svc.AssignWorkItemEx(wID, clientData);
    }

catch( Exception ex)
    {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Cancel Work Item

Description

Cancels a manual work item based on a specified manual work item identifier. Only the following manual work item status can transition to a Canceled status: Assigned, New, Pseudo, and Overdue.

Syntax

public virtual WFEvent CancelWorkItem(string wID)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string workItemID = … //

try
 {
    WFEvent evt = svc.CancelWorkItem(wID );
 }

catch( Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Cancel Work Item (Extended Method)

Description

Cancels a manual work item based on a specified manual work item identifier. This method also contains an added routine to track the method call duration. Only the following manual work item status can transition to a Canceled status: Assigned, New, Pseudo, and Overdue.

Syntax

public virtual WFEvent CancelWorkItemEx(string wID, string clientData)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.
clientData	string	A string that contains the client data. If clientData is null, the system will keep existing client data, otherwise the relevant data would be overwritten.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string workItemID = …// work item ID
string clientData = null;

try
 {
    WFEvent evt = svc. CancelWorkItemEx(workItemID,
 clientData);
 }

catch( Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Complete Work Item

Description

Marks a work item as completed.

Syntax

public virtual WFEvent CompleteWorkItem(string wID)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string workItemID = …// work item ID

try
 {
    WFEvent evt = svc.CompleteWorkItem(workItemID);
 }

catch( Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Complete Work Item (Extended Method)

Description

Marks a work item as completed with client data.

Syntax

public virtual WFEvent CompleteWorkItemEx(string wID, string clientData)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.
clientData	string	A string that contains the client data. If clientData is null, the system will keep existing client data, otherwise the relevant data would be overwritten.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string workItemID = …// work item ID
string clientData = null;

try
 {
    WFEvent evt = svc. CompleteWorkItemEx(workItemID,
 clientData);
 }

catch( Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
```

```
    }
```

Supported Versions

3.2.0.4 and higher

Create Linked Work Item

Description

Creates a manual work item that is linked to another manual work item. The work item you create does not depend on the completion of the work item to which it is linked. In other words, the original (source) work item can be marked as completed before new work item is completed.

Syntax

public virtual WFEvent CreateLinkedWorkItem(string sourceWorkItemID, string workToPerform, string userID, WFTimeDuration duration, string clientData)

Parameters

Name	Type	Description
sourceWorkItemID	string	A unique, 32-character ID for the original, or source, work item.
workToPerform	string	A string that specifies the work to perform.
userID	string	A string that contains a user ID for the user associated with the work item.
duration	WFTimeDuration	A time-duration object that specifies the duration setting of the work item.
clientData	string	A string that contains the client data. If clientData is null, the system will keep existing client data, otherwise the relevant data would be overwritten.

Output

None.

Example

```
IWFWorkflowService svc = base.GetWorkflowService();
// get existing work item
string workItemID = ..// for example,
"90CF843AC57644058A391FBFA030F607"

try
 {
    // Get the source WFManualWorkItem object
    WFManualWorkItem sourceWorkItem =
 svc.GetWorkItem(workItemID
    string workToPerform = sourceWorkItem.Name; //different
    workToPerform can be used if desired
    WFTimeDuration duration = new WFTimeDuration();
    duration.Length = "15"; //for example, 15 days
    duration.Unit = WFTimeUnit.DAY;
    string user = @"[DOMAIN NAME]\username"; //the participant
 of the
    linked work item
    WFEvent evt = svc.CreateLinkedWorkItem(
              sourceWorkItem.WorkItemID,
              workToPerform,
              user,
              duration,
              null);
 }

catch( Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Create Linked Work Item (Extended Method)

Description

Creates a manual work item that is linked to another manual work item. The extended parameter bDependent is used to specify the dependency between the original work item and the linked work item. If bDependent is false, the work items are independent, just like CreateLinkedWorkItem().If bDependent is true, the original (source) work item cannot be marked as completed before new work item has been completed.

Syntax

public virtual WFEvent CreateLinkedWorkItemEx(string sourceWorkItemID, string workToPerform, string userID, WFTimeDuration duration, string clientData, bool bDependent)

Parameters

Name	Type	Description
sourceWorkItemID	string	A unique, 32-character ID for the original, or source, work item.
workToPerform	string	A string that specifies the work to perform.
userID	string	A string that contains a user ID for the user associated with the work item.
duration	WFTimeDuration	A time-duration object that specifies the duration setting of the work item.
clientData	string	A string that contains the client data. If clientData is null, the system will keep existing client data, otherwise the relevant data would be overwritten.
bDependent	bool	If true: The source work item waits until the linked work item is either completed or canceled, before it can be completed/canceled. If false: The source work item can be completed/canceled regardless

Name	Type	Description
		of whether the linked work item is completed/canceled.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = base.GetWorkflowService();

// get existing work item
string workItemID = ..// for example,
 "90CF843AC57644058A391FBFA030F607"

try
 {
    // Get the source WFManualWorkItem object
    WFManualWorkItem sourceWorkItem =
 svc.GetWorkItem(workItemID
    string workToPerform = sourceWorkItem.Name; //different
    workToPerform can be used if desired
    WFTimeDuration duration = new WFTimeDuration("15",
 WFTimeUnit.DAY, false);
    string user = @"[DOMAIN NAME]\username"; //the participant
 of the
    linked work item
    WFEvent evt = svc.CreateLinkedWorkItem(
             sourceWorkItem.WorkItemID,
             workToPerform,
             user,
             duration,
             null,
             true);
 }

catch( Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

4.6 and above

Create Pseudo Work Item

Description

Creates a task by a specific AgileWork or other module that has the following characteristics:

- It does not have to be completed in order for a process to advance to the next steps.
- Unless specifically canceled, it remains active through the duration of the entire process, not just the duration of the AgileWork or other module that created it.

This provides a way for tasks to be included in a user's or manager's task list purely for monitoring purposes.

Syntax

public virtual WFEvent CreatePseudoWorkItem(string sourceWorkItemID, string workToPerform, string userID, WFTimeDuration duration, string clientData, bool reserved)

Parameters

Name	Type	Description
sourceWorkItemID	string	A unique, 32-character ID for the original, or source, work item.
workToPerform	string	A string that specifies the work to perform.
userID	string	A string that contains a user ID for the user associated with the work item.
duration	WFTimeDuration	A time-duration object that specifies the duration setting of the work item.
clientData	string	A string that contains the client data. If clientData is null, the system will keep existing client data, otherwise the relevant data would be overwritten.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = base.GetWorkflowService();

// get existing work item
string workItemID = ..// for example,
 "90CF843AC57644058A391FBFA030F607"

try
 {
    // Get the source WFManualWorkItem object
    WFManualWorkItem sourceWorkItem =
 svc.GetWorkItem(workItemID
    string workToPerform = sourceWorkItem.Name; //different
    workToPerform can be used if desired
    WFTimeDuration duration = new WFTimeDuration("15",
 WFTimeUnit.DAY, false);
    string user = @"[DOMAIN NAME]\username"; //the participant
 of the linked work item
    WFEvent evt = svc.CreatePseudoWorkItem (
              sourceWorkItem.WorkItemID,
              workToPerform,
              user,
              duration,
              null,
              false);
 }

catch( Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Create Work Item

Description

Creates a manual work item.

Syntax

public virtual WFEvent CreateWorkItem(string activityInstanceID, string workToPerform, string userID, WFTimeDuration duration, string clientData)

Parameters

Name	Type	Description
activityInstanceID	string	A string that contains the ID for an activity instance.
workToPerform	string	A string that specifies the work to perform.
userID	string	A string that contains a user ID for the user associated with the work item.
duration	WFTimeDuration	A time-duration object that specifies the duration setting of the work item.
clientData	string	A string that contains the client data. If clientData is null, the system will keep existing client data, otherwise the relevant data would be overwritten.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string activityInstanceID = … // for example,
 "0172460E0AF943C6A6520044452BCAB3";
string workToPerform = … // for example, "SubmitRequest";
//different workToPerform can be used if desired

WFTimeDuration duration = new WFTimeDuration("15",
 WFTimeUnit.DAY, true );// business time
string user = @"[DOMAIN NAME]\username"; //the participant of
 the linked work item
```

```
try
 {
    WFEventevt = svc.CreateWorkItem(activityInstanceID,
    workToPerform, user, duration, null);
 }

catch(Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Get Work Item

Description

Retrieves the manual work item object for a specified ID.

Syntax

public virtual WFManualWorkItem GetWorkItem(string wID)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.

Output

WFMaunalWorkItem object.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string workItemID = …// for example,
 "54A648A0A3004A02981E7F0848820FE7";
```

```
try
 {
    WFAutomaticWorkItem workItem =
 svc.GetWorkItem(workItemID);
    Console.WriteLine("{0}", workItem.Name);
 }

catch( Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Get Work List By User ID

Description

Retrieves a work item collection by specifying a user name and work item status.

Syntax

public virtual WFManualWorkItem[] GetWorkListByUserID(string username, string status)

Parameters

Name	Type	Description
userName	string	A string that contains the user name for the user. This member must be a registered AgilePoint user.
status	string	The status of the associated item.

Statuses

Status	Description
Assigned	A task that is assigned to the user.

Status	Description
Completed	The completed task.
Re-Assigned	A task that has been reassigned to another user.
Canceled	A task that has been canceled.
Overdue	A task that is overdue.

Output

Array of WFManualWorkItem objects.

Example

```
//Get all WFManualWorkItem assigned to user
IWFWorkflowService svc = GetWorkflowService();
string userID = …// for example, @"Demo3\Administrator";
string filterByStatus= string.format("{0};{1}",
WFManualWorkItem.ASSIGNED, WFManualWorkItem.OVERDUE);

try
 {
    WFManualWorkItem[] workItems =
 svc.GetWorkListByUserID(userID,
    filterByStatus);
    foreach (WFManualWorkItem workItem in workItems)
        {
        Console.WriteLine("{0}", workItem.ApplName);
        Console.WriteLine("{0}", workItem.AssignedDate);
        Console.WriteLine("{0}", workItem.DefName);
        Console.WriteLine("{0}", workItem.DueDate);
        }
 }

catch(Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Query Work List

Description

Retrieves a list of manual work items that match a specified query expression.

Syntax

public virtual WFManualWorkItem[] QueryWorkList(WFQueryExpr expr)

Parameters

Name	Type	Description
expr	WFQueryExpr	An object that represents the where clause of a SQL query expression.

Output

An array of WFManualWorkItem objects that contain the work item data.

Example

```
IWFWorkflowService svc = GetWorkflowService();
WFAny any = WFAny.Create(WFManualWorkItem.ASSIGNED);
WFQueryExpr expr = new
 WFQueryExpr("WF_MANUAL_WORKITEM.STATUS",
SQLExpr.EQ, any, true);

try
 {
    WFManualWorkItem[] workItems = svc.QueryWorkList(expr);

    // Iterating through the list of the ManualWorkItem
    foreach (WFManualWorkItem workItem in workItems)
        {
        Console.WriteLine("{0}", workItem.ApplName);
        Console.WriteLine("{0}", workItem.AssignedDate);
        Console.WriteLine("{0}", workItem.DefName);
        Console.WriteLine("{0}", workItem.DueDate);
        }
 }
```

```
catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Query Work List (Extended Method)

Description

Retrieves a list of manual work items that match a SQL statement.

Syntax

public virtual WFManualWorkItem[] QueryWorkListEx(string sql)

Parameters

Name	Type	Description
sql	string	A string that contains the where clause of the SQL statement you want to query.

Output

An array of WFManualWorkItem objects.

Example

```
IWFWorkflowService svc = GetWorkflowService();

//Query Expression string for Comparison of the WORK_ITEM_ID
string sql = "WORK_ITEM_ID in
 ('0006EE0244ED431CB93F6253060DD21F', …)";

try
 {
    // Calling the QueryWorkListEx API with argument sql query
 expression string.
```

```
    WFManualWorkItem[] workItems = _svc.QueryWorkListEx(sql);

    // Iterating through the list of the ManualWorkItem
    foreach (WFManualWorkItem workItem in workItems)
        {
        Console.WriteLine("{0}", workItem.ApplName);
        Console.WriteLine("{0}", workItem.AssignedDate);
        Console.WriteLine("{0}", workItem.DefName);
        Console.WriteLine("{0}", workItem.DueDate);
        }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Reassign Update Work Item

Description

Reassigns a manual work item to another participant.

Syntax

public virtual WFEvent ReassignUpdateWorkItem(string workItemID, string orginalUserID, string newAssigneeUserID, string clientData)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.
originalUserID	string	The user ID for the original user assigned the work item.
newAssigneeUserID	string	The User ID for the user to which you want to assign the work item.

Name	Type	Description
clientData	string	A string that contains the client data. If clientData is null, the system will keep existing client data, otherwise the relevant data would be overwritten.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string workItemID = …// "0006EE0244ED431CB93F6253060DD21F"; //
 Work item id
string originalUserID = …// @"[DOMAIN NAME]\[user name]"; //
 new userid
string newAssigneedUserID = …// @"[DOMAIN NAME]\[user
 name]"; // new userid

try
 {
    WFEvent evt = ReassignUpdateWorkItem(workItemID,
 originalUserID, newAssigneedUserID, null);
 }

catch( Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

4.6 and higher

Reassign Work Item

Description

Reassigns a work item to another participant, and update the user name.

Syntax

public virtual WFEvent ReassignWorkItem(string wID, string userID)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.
userID	string	A string that contains a user ID for the user associated with the work item.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string wID = …// "0006EE0244ED431CB93F6253060DD21F"; // Work
 item id
string newAssigneedUserID = …// @"[DOMAIN NAME]\[user
 name]"; // new
userid

try
 {
    WFEvent evt = svc.ReassignWorkItem(wID,
 newAssigneedUserID);
 }

catch(Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Reassign Work Item (Extended Method)

Description

Reassigns a work item to another participant, and update the user name. The extended method includes client data.

Syntax

public virtual WFEvent ReassignWorkItemEx(string wID, string userID, string clientData)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.
userID	string	A string that contains a user ID for the user associated with the work item.
clientData	string	A string that contains the client data. If clientData is null, the system will keep existing client data, otherwise the relevant data would be overwritten.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
string clientData= "<?xml version="1.0" … …";
string workItemID = .../// for example,
 "0006EE0244ED431CB93F6253060DD21F"; // Work item id
string newAssigneedUserID = …// for example @"[DOMAIN
 NAME]\[USER NAME]"; // new userid

try
 {
```

```
    WFEvent evt = _svc.ReassignWorkItemEx(workItemID,
    newAssigneedUserID, client);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Undo Assign Work Item

Description

Unassigns a work item that was previously assigned to a user. This method applies to work items that can be assigned to members of task groups, where a work item can be assigned to or claimed by any of a group of users.

Syntax

public virtual WFEvent UndoAssignWorkItem(string wID)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string workItemID = …// for example,
 "03ABD59A0EB74D7A8741709478E83877";
```

```
try
 {
    WFEvent evt = svc.UndoAssignWorkItem(workItemID);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Undo Assign Work Item (Extended Method)

Description

Unassigns a work item that was previously assigned to a user. This method applies to work items that can be assigned to members of task groups, where a work item can be assigned to or claimed by any of a group of users. The extended method includes client data.

Syntax

public virtual WFEvent UndoAssignWorkItemEx(string wID, string clientData)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.
clientData	string	A string that contains the client data. If clientData is null, the system will keep existing client data, otherwise the relevant data would be overwritten.

Output

WFEvent object that provides the status of the transaction. Possible statuses are Success, Failed, and Sent.

Example

```
string url = "http://[hostname]:[port]/
AgilePointServer";
string workItemID = …// for example,
"03ABD59A0EB74D7A8741709478E83877";
string clientData = …///

try
 {
    WFEvent evt = svc.UndoAssignWorkItemEx(workItemID,
 clientData);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Update Work Item

Description

Updates a manual work item or automatic work item.

Syntax

public virtual void UpdateWorkItem(string workItemID, NameValue[] attributes)

Parameters

Name	Type	Description
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.
attribute	NameValue	A NameValue array that contains the attributes that needs to be updated in the work item. For a manual work item, the following attributes can be updated: • NAME • ORIGINAL_USER_ID • CLIENT_DATA, POOL_ID • POOL_INFO • STATUS • USER_ID • PRIORITY • DUE_DATE For an automatic work item, the following attributes can be updated: • DUE_DATE • STATUS - if the value is Canceled, Completed, Overdue, Running, or Waiting. For an automatic work item, only DUE_DATE can be updated.

Output

None.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string workItemID = …// work item ID of manual work item or
 automatic work item.

try
 {
    NameValue[] attrs = NameValue.Array(
    "NAME", "[New Name]",
    "DUE_DATE", [DateTime]); // for example,
 DateTime.Now.AddDays(3.0)
    svc.UpdateWorkItem(workItemID, attrs );
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4and higher

Methods for Notifications

This section describes service calls related to email notifications.

Cancel Mail Deliverable

Description

Cancels the failed mail deliverable record based on a given message identifier. Note that canceling the failed mail deliverable record prevents it from being recycled or present on a given interval by the AgilePoint engine.

Syntax

public virtual void CancelMailDeliverable(string mID)

Parameters

Name	Type	Description
mID	string	A string that contains the unique ID for an email notification.

Output

None.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string emailNotificatioID = …

try
    {
    svc.CancelMailDeliverable(emailNotificatioID);
    }

catch (Exception ex)
    {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Get Expecting Send Mail Deliverable

Description

Retrieves all the failed sent and scheduled to resend email notifications.

Syntax

public virtual WFMailDeliverable[] GetExpectingSendMailDeliverable()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

Array of WFMailDeliverable objects.

Example

```
IWFWorkflowService svc = GetWorkflowService();

try
 {
    //Returns Array of WFMailDeliverable type
    WFMailDeliverable[] mailDeliverables =
    svc.GetExpectingSendMailDeliverable();
    foreach(WFMailDeliverable m in mailDeliverables)
    {
       Console.WriteLine("Mail ID: '{0}'", m.ID);
       Console.WriteLine("Process Instance ID:
 '{0}'",m.ProcInstID);
       Console.WriteLine("E-Mail Subject: '{0}'",
 m.Mail.Subject);
    }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Get Mail Deliverables

Description

Retrieves all the mail deliverables for a process instance.

Syntax

public virtual WFMailDeliverable[] GetMailDeliverables(string piID)

Parameters

Name	Type	Description
processInstanceID	string	A string that contains the ID of the process instance.

Output

Array of WFMailDeliverable objects.

Example

```
//Process Instance ID associated with the Process Instance.
IWFWorkflowService svc = GetWorkflowService();
string processInstanceID = …// for example,
 "1e3d514d43d3465cae6ec3bbbd409168";

try
 {
    WFMailDeliverable[]emailNotifications =
    svc.GetMailDeliverables(processInstanceID);
    foreach(WFMailDeliverable m in emailNotifications)
     {
        Console.WriteLine("Mail ID: '{0}'", m.ID);
        Console.WriteLine("Process Instance ID: '{0}'",
 m.ProcInstID);
        Console.WriteLine("E-Mail Subject: '{0}'",
 m.Mail.Subject);
     }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

Resend Mail Deliverable

Description

Resends the mail deliverable with a specified mail ID.

Syntax

public virtual void ResendMailDeliverable(string mID)

Parameters

Name	Type	Description
mID	string	A string that contains the unique ID for an email notification.

Output

None.

Example

```
//Sample for using resendMailDeliverable
IWFWorkflowService svc = GetWorkflowService();
string emailNotificationID = … // for example,
"149C3974240F47D3B28EB6D4A3CDCD3F"

try
 {
    svc.ResendMailDeliverable(emailNotificationID);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

Methods for Events

This section describes service calls related to workflow events.

Get Event

Description

Retrieves an event object. This service call is usually used to check if a service call has been completed.

Syntax

public virtual WFEvent GetEvent(string evtID)

Parameters

Name	Type	Description
eventID	string	A unique, 32-character event ID.

Output

WFEvent object.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string eventID = …// for example,
 "049C3974240F47D3BA8EB6D4A3CDCD3F";

try
 {
    WFEvent evt = _workflowAPI.GetEvent(eventID);
    Console.WriteLine("Event ID: '{0}'", evt.EventID);
    Console.WriteLine("Event Name: '{0}'",evt.EventName);
    Console.WriteLine("Event Status: '{0}'",evt.Status);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
```

```
}
```

Supported Versions

3.2.0.4 and higher

Methods for Custom Attributes

This section describes service calls related to custom attributes.

Get Custom Attribute

Description

Retrieves a single custom attribute.

Syntax

public virtual object GetCustomAttr(string customID, string name);

Parameters

Name	Type	Description
customID	string	A work object ID as identified in the process instance. This method allows only one custom ID (work object ID).
name	string	A string that contains the name of an item, such as a property or attribute.

Output

Custom attribute value (can be string, integer, float, double, bool, and/or DateTime).

Example

```
IWFWorkflowService svc = GetWorkflowService();
```

```
string customID = …// for example,
 "013933F128C3415F81D6F545594D4CB6";
string name = …// for example,"/pd:myFields/pd:Name" or
 "Approval"

try
 {
    Object obj = svc.GetCustomAttr(customID,name);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Get Custom Attributes

Description

Retrieves a collection of custom attributes for a specified custom ID in XML format.

Syntax

public virtual string GetCustomAttrs(string customID)

Parameters

Name	Type	Description
customID	string	A work object ID as identified in the process instance. This method allows only one custom ID (work object ID).

Output

string that contains all the attributes for the custom ID in XML format.

Example

```
// get custom attributes in xml format
IWFWorkflowService svc = GetWorkflowService();
string customID = …// for example,
 "013933F128C3415F81D6F545594D4CB6";

try
 {
    string xml = svc.GetCustomAttrs(sessionID, string
 processInstanceID);
    Console.WriteLine("AttributeXMLstring={0}",resultAttrXML);
    WFCustomAttributes attrs = new WFCustomAttributes();
    attrs.AttrXml = xml; // de-serialize xml
    string[] attributeNames = attrs.GetNames();// get
 attribute names
    Object value = attrs["MyAttributeName"]; // retrieve
 attribute value
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Get Custom Attributes (Extended Method)

Description

Retrieves all custom attributes for a specified a set of custom IDs.

Syntax

public virtual KeyValue[] GetCustomAttrsEx(string[] customIDs)

Parameters

Name	Type	Description
customIDs	string	An array of strings that contains custom IDs (work object IDs).

Output

An array of KeyValue values. The key is a custom ID, and value is a string of custom attributes in XML format.

Example

```
IWFWorkflowService svc = GetWorkflowService();

//Array of custom ID
string[] customIDs = …// for example,
 {"InfoPath:011eaf6c46ac4723b25b4db5772d9912", ….};

try
 {
    KeyValue[] keyValues = svc.GetCustomAttrsEx(customIDs);
    foreach (KeyValue kv in keyValues)
        {
        Console.WriteLine("ID:'{0}'", kv.Key);
        Console.WriteLine("Custom Attributes XML: '{0}'",
 kv.Value);
        WFCustomAttributes attrs = new WFCustomAttributes();
        Attrs.AttrXml = kv.Value;
        }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Remove Custom Attribute

Description

Removes a custom attribute from a custom ID.

Syntax

public virtual void RemoveCustomAttr(string customID, string attributeName)

Parameters

Name	Type	Description
customID	string	A work object ID as identified in the process instance. This method allows only one custom ID (work object ID).
name	string	A string that contains the name of an item, such as a property or attribute.

Output

None.

Example

```
//Sample for using Workflow.RemoveCustomAttr
IWFWorkflowService svc = GetWorkflowService();
string customID = ..// for example,
 "InfoPath:011eaf6c46ac4723b25b4db5772d9912"
string attributeName = ...// for example, "//pd:purchaseOrder/
pd:secondApproval"

try
 {
    svc.RemoveCustomAttr(customID, attributeName);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
```

```
}
```

Supported Versions

3.2.0.4 and higher

Set Custom Attribute

Description

Sets the name and value for a custom attribute for a specified custom ID.

Syntax

public virtual SetCustomAttr(String customID, String name, object val)

Parameters

Name	Type	Description
customID	string	A work object ID as identified in the process instance. This method allows only one custom ID (work object ID).
name	string	A string that contains the name of an item, such as a property or attribute.
val	object	An object that contains the value of the custom attribute.

Output

None.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string customID = ..// for example,
"InfoPath:011eaf6c46ac4723b25b4db5772d9912"
string attributeName = ...// for example,
"//pd:purchaseOrder/pd:secondApproval"
```

```
bool attributeValue = true;

try
 {
    svc.SetCustomAttr(customID, attributeName,
 attributeValue);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Set Custom Attributes

Description

Sets names and values for multiple custom attributes for a specified custom ID.

Syntax

public virtual void SetCustomAttrs(string customID, NameValue[] nameValues)

Parameters

Name	Type	Description
customID	string	A work object ID as identified in the process instance. This method allows only one custom ID (work object ID).
attributes	NameValue	Name-value pairs associated with a custom ID.

Output

None.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string customID = ..// for example,
 "InfoPath:011eaf6c46ac4723b25b4db5772d9912"

NameValue[] attributes = new NameValue[]
    {
    new
 NameValue("CustomAttributeName1","CustomAttributevalue1"),
    new NameValue("CustomAttributeName2",false),
    new NameValue("CustomAttributeName3", 10.0)
 };

try
 {
    svc.SetCustomAttrs(customID, attributes);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Methods for Archiving and Restoring Processes

This section describes service calls related to archiving and restoration of processes.

Archive Process Instance

Description

Archives a process instance based on a specified process instance identifier by moving the set of process instance records from the current AgilePoint Database into the AgilePoint Archive Database. The process instance records and all of the associated data are then deleted from the AgilePoint Database. The process instance to be archived must be completed or canceled.

Syntax

public virtual void ArchiveProcInst(string procInstID)

Parameters

Name	Type	Description
processInstanceID	string	A string that contains the ID of the process instance.

Output

None.

Example

```
// this is console application sample
IWFWorkflowService svc = GetWorkflowService();
string processInstanceID = ..// the ID of process instance to
 be
     archived

try
    {
    svc.ArchiveProcInst(processInstanceID);
    }

catch (Exception ex)
    {
    Console.WriteLine("Failed!, {0}",
 ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Restore Process Instance

Description

Restores a process instance and associated data from the ArchiveDatabase to the AgilePoint Server. The process instance records are written to the AgilePoint Database deleted from the AgilePoint Archive Database.

Syntax

public virtual void RestoreProcInst(string procInstID)

Parameters

Name	Type	Description
processInstanceID	string	A string that contains the ID of the process instance.

Output

None.

Example

```
// this is console application sample code
IWFWorkflowService svc = GetWorkflowService();
string processInstanceID = ..// the ID of process instance to
 be
restored.

try
 {
    svc.RestoreProcInst(processInstanceID);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " +
 ShUtil.GetSoapMessage(ex) );
 }
```

Supported Versions

3.2.0.4 and higher

Query Archived Process Instances

Description

Retrieves process instances that match a SQL query.

Syntax

public virtual WFBaseProcessInstance[] QueryArchivedProcInsts(string sql)

Parameters

Name	Type	Description
sql	string	A string that contains the where clause of the SQL statement you want to query.

Output

An array of WFBaseProcessInstance objects.

Example

```
// Build SQL Statement
string processDefinitionName = "Budget Request Approval
 Process";
string sqlWhereClause = string.Format("DEF_NAME ='{0}'",
processDefinitionName);

try
 {
    WFBaseProcessInstance[] archivedProcessInstances =
    svc.QueryArchivedProcInsts(sqlWhereClause)
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Send Mail

This section describes service calls related to sending email using AgilePoint.

Send Mail

Description

Sends an email through AgilePoint Server.

Syntax

public virtual void SendMail(String to, String cc, String subject, String body)

Parameters

Name	Type	Description
to	string	Recipient of the email.
cc	string	Persons copied on the email.
subject	string	The subject of the email.
body	string	The text of the email (body).

Output

None.

Example

```
//Sample for using Workflow.SendMail
IWFWorkflowService svc = GetWorkflowService();

try
 {
    // email recipients
```

```
    string to = "bill@tusca.com";

    // CC
    string cc = "bob@tusca.com";

    //Subject of the Mail
    string subject = "This is email Subject";

    //Body of the Mail
    string body = "This email Body";
    svc.SendMail(to, cc, subject, body);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Send Mail (Extended Method)

Description

Sends an email through AgilePoint Server. The extended method enables you to send attachments.

Syntax

public virtual void SendMailEx(String from, String to, String cc, String subject, String body , String attachments)

Parameters

Name	Type	Description
from	string	The sender of the email.
to	string	Recipient of the email.
cc	string	Persons copied on the email.
subject	string	The subject of the email.
attachments	string	File attachments included with the email. This parameter

Name	Type	Description
		must use a file path from the file system (for example, C:\file.txt) on the machine where AgilePoint Server is installed. If there is no attachment, you can pass **null** or **String.Empty**.

Output

None.

Example

```
//Sample for using Workflow.SendMailEx
IWFWorkflowService svc = GetWorkflowService();

try
 {
    string from = "john@tusca.com"; // or <Full Name>"email
 address"
    string to = "bill@tusca.com";
    string cc = "bob@tusca.com";
    string subject = "Mail Subject";
    string body = "Mail Body";
    string attachments = "c:\\Temp\\Tempdoc.doc";

    //Send Mail
    svc.SendMailEx(from, to, cc, subject, body, attachments);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

4.6 and higher

Send Mail (Extended Method with Priority)

Description

Sends an email through AgilePoint Server. The extended method enables you to send attachments.

Syntax

public virtual void SendMailEx(String from, String to, String cc, String subject, String body , String attachments, Enum priority)

Parameters

Name	Type	Description
from	string	The sender of the email.
to	string	Recipient of the email.
cc	string	Persons copied on the email.
subject	string	The subject of the email.
attachments	string	File attachments included with the email. This parameter must use a file path from the file system (for example, C:\file.txt) on the machine where AgilePoint Server is installed. If there is no attachment, you can pass **null** or **String.Empty**.
priority	enum	The email's priority. Values can be **high**, **normal**, or **low**.

Output

None.

Example

```
//Sample for using Workflow.SendMailEx
IWFWorkflowService svc = GetWorkflowService();

try
 {
    string from = "john@tusca.com"; // or <Full Name>"email
 address"
    string to = "bill@tusca.com";
    string cc = "bob@tusca.com";
    string subject = "Mail Subject";
    string body = "Mail Body";
    string attachments = "c:\\Temp\\Tempdoc.doc";
    enum priority = "high";
```

```
    //Send Mail
    svc.SendMailEx(from, to, cc, subject, body, attachments);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

5.0 and higher

Other Web Services

This section describes other additional web services.

Query Audit Trail

Description

Retrieves all audit trail items.

Syntax

public virtual WFAuditTrailItem[] QueryAuditTrail(string where)

Parameters

Name	Type	Description
sql	string	A string that contains the where clause of the SQL statement you want to query.

Output

An array list of WFAuditTrailItem objects.

Example

```
// This is console application sample
IWFWorkflowService svc = GetWorkflowService();

// Checking where condition based on the CATEGORY COLUMNand
 PURPOSE
COLUMN .
string where = "CATEGORY = 0 AND PURPOSE='Check-in process
definition'";

try
 {
    // calling the QueryAuditTrail web method it return a
 array of Dataset.
    WFAuditTrailItem[] result = svc.QueryAuditTrail(where);
    foreach (WFAuditTrailItem item in result)// Iterating
 through
    WFAuditTrailItem
    {
        //Displaying the result on the console.
        System.Console.Write("ItemCategory:
 {0} ,ItemDateOccurred: {1}",
                    item.Category,item.DateOccurred);
        System.Console.Write("ItemDescription:
 {0} ,ItemObjectID: {1}",
                    item.Description, item.ObjectID);
        System.Console.WriteLine("ItemPurpose:
 {0} ,ItemPerformer: {1}",
                    item.Purpose item.Performer);
        }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

4.0.1 and higher

Query Database

Description

Queries the database with any valid sql query and returns the dataset as a string in XML format.

Syntax

public virtual string QueryDatabase(string sql)

Parameters

Name	Type	Description
sql	string	A string that contains the where clause of the SQL statement you want to query.

Output

An XML string that contains the dataset with the results of the database query.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string sql = "SELECT * FROM WF_AUDIT_TRAILS where CATEGORY = 0
 AND
PURPOSE='Check-in process definition'";

try
 {
    // calling the QueryDatabase webmethod and passing the sql
 query as the argument.
    string xml = svc.QueryDatabase(sql);
    Console.WriteLine("{0}", xml);// Displaying the XML string
 on console.
    System.IO.stringReader sr = new
 System.IO.stringBuilder(xml);
    System.Data.DataSet ds = new System.Data.DataSet();
    ds.LoadXml(sr);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Query Database (Extended Method)

Description

Queries the database with any valid SQL query and returns an array of 2 elements in XML string format.

Syntax

public virtual string[] QueryDatabaseEx(string sql)

Parameters

Name	Type	Description
sql	string	A string that contains the where clause of the SQL statement you want to query.

Output

2 elements in a string array, where string[0] is an XML string with the dataset of the query results and string[1] is an XML string with the schema.

Example

```
IWFWorkflowService svc = GetWorkflowService();
string sql = "SELECT * FROM WF_AUDIT_TRAILS where CATEGORY = 0
 AND
PURPOSE='Check-in process definition'";

try
 {
    // calling the QueryDatabase webmethod and passing the sql
 query as the argument.
    string[] xmls = svc.QueryDatabase(sql);
    Console.WriteLine("{0}", xmls[0]);// Displaying the XML
 string on console.
    Console.WriteLine("{0}", xmls[1]);// Displaying the schema
 on console.
 }

catch (Exception ex)
 {
```

```
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
}
```

Supported Versions

3.2.0.4 and higher

Administrative Service

This section describes service calls related to administrative functions. These calls are found under admin.wsdl.

Get Database Information

Description

Retrieves the database information of the current server configuration.

Syntax

public virtual DatabaseInfo GetDatabaseInfo()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

A DatabaseInfo object that represents the database information of the system.

Example

```
IWFAdminService svc = GetAdminService();

try
 {
    DatabaseInfo dbInfo = svc.GetDatabaseInfo();

    if (dbInfo != null)
```

```
    {
       Console.WriteLine("AgilePoint System Database
 Information:");
       Console.WriteLine("Vendor: {0}", dbInfo.Vendor);
       Console.WriteLine("Provider: {0}", dbInfo.Provider);
       Console.WriteLine("Connection string: {0}",
 dbInfo.Connstr);
    }
    }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
/*
This example produces the following results:
AgilePoint System Database Information:
Vendor: MSSQLDatabase
Provider:
Connection string: application name=AgilePoint
 Server;connection
lifetime=5;min pool
size=10;server=VIT-34;database=AgilePointDB;trusted_Connection=yes
*/
```

Supported Versions

4.6 and higher

Get Domain Groups

Description

Retrieves all the domain group objects.

Syntax

public virtual KeyValue[] GetDomainGroups(string source, string filter)

Parameters

Name	Type	Description
source	string	A string that contains the LDAP path to the domain. If the value

Name	Type	Description
		is null, the AgilePoint Server machine domain will be used.
filter	string	A string that contains the wildcard-enabled group filter in the format (name=<<wildcard filter>>)

Output

Array of KeyValue objects. If the specified group is not found, the output is null.

Example

```
IWFAdminService svc = GetAdminService();
string activeDirectoryLdapPath = ...// for example, LDAP://
ou=Sales,dc=Frabrikam,dc=com
string groupNameFilter = "A*";
groupNameFilter = string.Format("(name={0})",
 groupNameFilter);

try
 {
    KeyValue[] grps =
 svc.GetDomainGroups(activeDirectoryLdapPath,
 groupNameFilter);
 foreach (KeyValue grp in grps)
        {
        Console.WriteLine("Group Name:{0}; Group Distinct
 Name:{1};", grp.Key, grp.Value);
        }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Get Domain Group Members

Description

Retrieves the members of a domain group.

Syntax

public virtual DomainUser[] GetDomainGroupMembers(string groupDistinguishedName)

Parameters

Name	Type	Description
groupDistinguishedName	string	A string that contains the name of the group in LDAP format.

Output

An array of DomainUser objects.

Example

```
IWFAdminService svc = GetAdminService();
string groupName = "Administrators";
string distinguishedGroupName = "LDAP://" + groupName;

try
 {
    DomainUser[] grpUsers = svc.
    GetDomainGroupMembers(distinguishedGroupName);
    foreach (DomainUser usr in grpUsers)
        {
        Console.WriteLine("User Name:{0}; Full Name:{1};",
 usr.UserName, usr.FullName);
        }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
/*
This example produces the following results:
```

```
User Name:Administrator; Full Name:Administrator
User Name:vitbdc/yuvarajn; Full Name:Yuvaraj Nagarajan
*/
```

Supported Versions

3.2.0.4 and higher

Get Domain Name

Description

Retrieves the domain name to which AgilePoint Server connects.

Syntax

public virtual string GetDomainName()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

A string containing the domain name of the AgilePoint Server machine.

Example

```
IWFAdminService svc = GetAdminService();

try
 {
    string domainName = svc.GetDomainName();
    Console.WriteLine("AgilePoint System Domain Name={0}",
 domainName);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

```
/* Sample of output
AgilePoint System Domain Name=LDAP://dc=Frabrikam,dc=com
*/
```

Supported Versions

4.6 and higher

Get Domain Users

Description

Retrieves all the user information in the domain that AgilePoint Server connects. It could be a local Windows system user, or a domain controller on the network.

Syntax

public virtual DomainUser[] GetDomainUsers(string source, string filter)

Parameters

Name	Type	Description
source	string	A string that contains the LDAP path to the domain. If the value is null, the AgilePoint Server machine domain will be used.
filter	string	A string that contains the wildcard-enabled group filter in the format (name=<<wildcard filter>>)

Output

Array of DomainUser objects. If the specified group does not exist, the return value is null.

Example

```
IWFAdminService svc = GetAdminService();
string activeDirectoryLdapPath = ...// for example,
LDAP://ou=Sales,dc=Frabrikam,dc=com
```

```
string userFilter = ""; // All Users

try
 {
    DomainUser[] users =
 svc.GetDomainUsers(activeDirectoryLdapPath, userFilter);
    foreach (DomainUser user in users)
    {
       Console.WriteLine("Full Name:{0}; Login Name:{1};",
 user.FullName, user.UserName);
    }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }

/*
This example produces the following results:
Full Name:Manager; Login Name:VIT-34\Manager;
Full Name:Marcomm; Login Name:VIT-34\Marcomm;
Full Name:SharePoint Administrator; Login Name:VIT-34\sp_adm;
Full Name:Sujeet Kumar; Login Name:VIT-34\sujeetk;
*/
```

Supported Versions

3.2.0.4 and higher

Get Locale

Description

Retrieves the default locale for the AgilePoint Server.

Syntax

public virtual string GetLocale()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

Locale abbreviation—for example, en-US.

Example

```
IWFAdminService svc = GetAdminService();
string activeDirectoryLdapPath = ...// for example,
LDAP://ou=Sales,dc=Frabrikam,dc=com
string userFilter = ""; // All Users

try
 {
    string locale = svc.GetLocale();
    Console.WriteLine("Locale = '{0}'", locale);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }

/*
This example produces the following results:
Locale = 'en-US
*/
```

Supported Versions

4.6 and higher

Get Register User

Description

Retrieves the user information for the registered user.

Syntax

public virtual RegisteredUser GetRegisterUser(string userName)

Parameters

Name	Type	Description
userName	string	A string that contains the user name for the user. This member must be a registered AgilePoint user.

Output

RegisteredUser object.

Example

```
IWFAdminService svc = GetAdminService();

try
 {
    RegisteredUser registerUser =
    svc.GetRegisterUser("domain\\wilson.goodman");
    Console.WriteLine("Name = '{0}', Department = '{1}'",
                   registerUser.FullName,
                   registerUser.Department);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }

//This example produces the following results:
//Name = 'CFO' Department = 'Finance'
```

Supported Versions

3.2.0.4 and higher

Get Register Users

Description

Retrieves all registered users.

Syntax

public virtual RegisteredUser[] GetRegisterUsers()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

Returns an array of registered users.

Example

```
IWFAdminService svc = GetAdminService();

try
 {
    RegisteredUser[] registerUsers = svc.GetRegisterUsers();
    foreach (RegisteredUser regUser in registerUsers)
    {
       Console.WriteLine("Name = '{0}', Department = '{1}'",
                   registerUser.FullName,
                   registerUser.Department);
    }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Get Register User Icons

Description

Retrieves an icon for a registered user.

Syntax

public virtual byte[] GetRegisteredUserIcon(string userName)

Parameters

Name	Type	Description
userName	string	A string that contains the user name for the user. This member must be a registered AgilePoint user.

Output

Return an array of bytes that contains image data.

Example

```
For future use.
```

Supported Versions

4.6 and higher

Get Sender Email Address

Description

Retrieves the sender email address of the AgilePoint Server.

Syntax

public virtual string GetSenderEMailAddress()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

The email address that is configured as the sender email address on the AgilePoint Server.

Example

```
IWFAdminService svc = GetAdminService();

try
 {
    string senderEMailAddress = svc.GetSenderEMailAddress();
    Console.WriteLine("Sender EMail Address = '{0}'",
    senderEMailAddress);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }

//This example produces the following results:
//Sender EMail Address = 'admin@your-domain.com'
```

Supported Versions

4.6 and higher

Get SMTP Server

Description

Retrieves the SMTP server of the current server configuration.

Syntax

public virtual string GetSmtpServer();

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

A string that contains the name of the AgilePoint system's SMTP server.

Example

```
IWFAdminService svc = GetAdminService();

try
 {
    string smtpServer = svc.GetSmtpServer();
    Console.WriteLine("SMTP Server Name = '" + smtpServer +
 "'");
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }

//This example produces the following results:
//SMTP Server Name = 'smtp.vitinfotech.com'
```

Supported Versions

4.6 and higher

Get System Performance Information

Description

Retrieves system performance information for AgilePoint Server.

Syntax

public virtual WFSysPerfInfo GetSysPerfInfo()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

WFSysPerfInfo object.

Example

```
IWFAdminService svc = GetAdminService();

try
 {
    WFSysPerfInfo sysPerfInfo = svc.GetSysPerfInfo();
    Console.WriteLine("1) ServerID =
 '{0}'",sysPerfInfo.ServerID);
    Console.WriteLine("2) MemoryAllocated = '{0}'",
    sysPerfInfo.MemoryAllocated);
    //...
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }

/*
This example produces the following results:
1) ServerID = 'DEMO/4356'
2) MemoryAllocated = '7329'
*/
```

Supported Versions

3.2.0.4 and higher

Get System User

Description

Retrieves the name of the system user.

Syntax

public virtual string GetSystemUser()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

Returns the name of the system user as a string value.

Example

```
IWFAdminService svc = GetAdminService();

try
 {
    string systemUser = svc.GetSystemUser();
    Console.WriteLine("AgilePoint System User= '{0}'",
 systemUser);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }

//This example produces the following results:
//AgilePoint System User = 'Administrator'
```

Supported Versions

4.6 and higher

Query Register Users

Description

Retrieves the list of registered users on theAgilePoint Server.

Syntax

public virtual RegisteredUser[] QueryRegisterUsers(string sql)

Parameters

Name	Type	Description
sql	string	A string that contains the where clause of the SQL statement you want to query.

Output

Returns a list of registered users.

Example

```
IWFAdminService svc = GetAdminService();
string sqlWhereClause = "DEPARTMENT in ('PublicUsers', ...)"

try
 {
    RegisteredUser[] registeredUsers =
    adminService.QueryRegisterUsers(sqlWhereClause);
    foreach (RegisteredUser user in registeredUsers)
    {
       Console.WriteLine("User Name = '{0}', Email = '{1}'",
       user.UserName,user.EMailAddress);
    }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }

/*
```

```
This example produces the following results:
User Name = 'DEMO\Author' Email = 'author@DEMO.com'
User Name = 'DEMO\Employee' Email = 'employee@DEMO.com'
*/
```

Supported Versions

Register User

Description

Registers a user on the AgilePoint system.

Syntax

public virtual void RegisterUser(RegisteredUser user)

Parameters

Name	Type	Description
user	RegisteredUser	A RegisteredUser object that contains user information.

Output

None.

Example

```
IWFAdminService svc = GetAdminService();

try
 {
    RegisteredUser newUser= new RegisteredUser();
    newUser.FullName = "Accountant";
    newUser.UserName = "DEMO\\Accountant";
    newUser.Department = "Accounts";
    newUser.EMailAddress = "accountant@DEMO.com";
    newUser.Locale = "en-US";
    svc.RegisterUser(newUser);
 }
```

```
catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Unregister User

Description

Removes a user's registration from the AgilePoint system. Note that this call does not remove the user from the local Windows system or the domain controller.

Syntax

void UnregisterUser(string userName)

Parameters

Name	Type	Description
userName	string	A string that contains the user name for the user. This member must be a registered AgilePoint user.

Output

None.

Example

```
IWFAdminService svc = GetAdminService();
string userName = ...// for example, "[Domain Name]\[User
 Account
Name]"

try
 {
```

```
svc.UnregisterUser(userName);
 }

catch( Exception ex)
 {
Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Update Registered User

Description

Updates user data for a registered user.

Syntax

public virtual void UpdateRegisterUser(RegisteredUser user)

Parameters

Name	Type	Description
user	RegisteredUser	A RegisteredUser object that contains user information.

Output

None.

Example

```
RegisteredUser user = new RegisteredUser();
user.UserName = "DEMO3\\cfo";
user.FullName = "Andy";
user.EMailAddress = "cfo@tusca.com";
user.Title = "CFO";
user.Department = "Marketing";
user.Manager = "DEMO3\\Administrator";
user.Locale = Thread.CurrentThread.CurrentUICulture.Name;
```

```
IWFAdminService svc = GetAdminService();

try
 {
    svc.UpdateRegisterUser( user );
 }

catch( Exception ex )
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Update Registered User Icon

Description

Updates the icon for a registered user.

Syntax

public virtual void UpdateRegisteredUserIcon(string userName, byte[] userIcon)

Parameters

Name	Type	Description
userName	string	A string that contains the user name for the user. This member must be a registered AgilePoint user.
userIcon	byte	Byte array representation of icon image.

Output

None.

Example

```
None.
```

Supported Versions

4.6 and higher

Group, Role, and Rights

This section describes service calls related to the groups, roles, and rights.

Add Group

Description

Adds a group to the AgilePoint system.

Syntax

public virtual WFGroup AddGroup(string groupName, string description, string responsibleUser, bool enabled)

Parameters

Name	Type	Description
groupName	string	A string that contains the name of the group.
description	string	A string that contains the description of an entity.
responsibleUser	string	A string that contains the name of the responsible user of this group. The responsible user must be a registered AgilePoint user.

Name	Type	Description
enabled	bool	A Boolean value: True enables the entity; False disables the entity.

Output

WFGroup object represented the group that is added.

Example

```
IWFAdminService svc = GetAdminService();
string groupName = ...;
string description = ...;
string responsibleUser = @"[Domain Name]\[Account Name]", //
 Group
Lead User Name
bool enable = true;

try
    {
    WFGroup group = svc.AddGroup(groupName, description,
    responsibleUser, enable);
    }

catch( Exception ex )
    {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Add Group Member

Description

Adds a user as a member of a group.

Syntax

public virtual WFGroupMember AddGroupMember(string groupName, string userName, string description, string clientData, bool enabled)

Parameters

Name	Type	Description
groupName	string	A string that contains the name of the group.
userName	string	A string that contains the user name for the user. This member must be a registered AgilePoint user.
description	string	A string that contains the description of an entity.
clientData	string	A string that contains the client data. If clientData is null, the system will keep existing client data, otherwise the relevant data would be overwritten.
enabled	bool	A Boolean value: True enables the entity; False disables the entity.

Output

WFGroupMember object that contains the data for the new group member.

Example

```
IWFAdminService svc = GetAdminService();
string groupName = ...;
string description = ...;
string userName = @"[Domain Name]\[Account Name]", // Group
 Lead User Name
string clientData = null;
bool enable = true;

try
    {
    WFGroup group = svc.AddGroupMember(groupName, userName,
 description,
    clientData, enable);
    }

catch( Exception ex )
    {
```

```
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Add Role

Description

Adds a role to the AgilePoint system.

Syntax

public virtual WFRole AddRole(String roleName, String description, int[] rights, bool enabled)

Parameters

Name	Type	Description
roleName	string	A string that contains the name of a role.
description	string	A string that contains the description of an entity.
rights	WFAccessRights	Array of index values that specifies the rights that are assigned to the Role. See the following table for appropriate indexes. WFAccessRights provides the enums for rights. See the sample code for more information.
enabled	bool	A Boolean value: True enables the entity; False disables the entity.

Access Rights

Index	Rights
0	Register and modify the user information
1	Unregister a user
2	Add and modify role information
3	Remove a role
4	Add and modify group information
5	Remove a group
6	Modify and view system information
7	Add a process definition
8	Check in and check out a process definition
9	Delete or disable a process definition
10	Release a process definition
11	Initiate a process
12	Suspend and resume a process
13	Resend and cancel an email notification
14	Cancel a process
15	Rollback a process
16	Reassign a task
17	Cancel a task
18	Create a task
19	Add, remove and modify delegation
20	Add, remove and modify report configuration
21	Achieve and restore processes
22	Add, remove and modify shared custom attributes
23	View process details

Output

WFRole object for the role that is added.

Example

```
IWFAdminService svc = GetAdminService();
string roleName = ...// for example, "Process Manager";
string description = ...;
/*integer array specifying the access rights for this role.
* 11 - Initiate a process
* 14 - Cancel a process
* 23 - view process details
*/
int[] rights =
    {
    WFAccessRights.InitiateProcessInstance,
    WFAccessRights.CancelProcessInstance,
    WFAccessRights.ViewProcessDetails
 };

try
 {
    WFRole role = svc.AddRole(roleName, description, rights,
 true);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Add Role Member

Description

Adds a user or a group to a role.

Syntax

public virtual WFRoleMember AddRoleMember(string roleName, string assignee, string assigneeType, string clientData, string objectID, string objectType);

Parameters

Name	Type	Description
roleName	string	A string that contains the name of a role.
assignee	string	The registered user name or group name.
assigneeType	string	The type for the assignee: Either user or group.
clientData	string	A string that contains the client data. If clientData is null, the system will keep existing client data, otherwise the relevant data would be overwritten.
objectID	string	Reserved for future use. Leave null.

Output

WFRoleMember object.

Example

```
IWFAdminService svc =
 GetAdminService();

string roleName = "Administrative";
string assignee = @"Demo3\Administrator";
string assigneeType = "User";

try
    {
    WFRoleMember member = svc.AddRoleMember(roleName, assign,
    assignType, "", null, null);
    }

catch (Exception ex)
    {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Enabled Group Member

Description

Enables or disables a user as a member of a group.

Syntax

public virtual WFGroupMember EnabledGroupMember(string groupName, string userName, bool enabled)

Parameters

Name	Type	Description
groupName	string	A string that contains the name of the group.
userName	string	A string that contains the user name for the user. This member must be a registered AgilePoint user.
enabled	bool	A Boolean value: True enables the entity; False disables the entity.

Output

WFGroupMember object.

Example

```
IWFAdminService svc =
 GetAdminService();

string groupName = "Administrative";
string userName = @"Demo3\Administrator";
bool enabled =false;
```

```
try
 {
    svc.EnabledGroupMember(groupName, userName, enabled);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Get Access Right Names

Description

Retrieves the names of all the access rights in the AgilePoint system.

Syntax

public virtual string[] GetAccessRightNames()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

An array of strings that contain the names of all the access rights for the system.

Example

```
IWFAdminService svc = GetAdminService();

try
 {
    string[] permissionNames = svc. GetAccessRightNames();
 }
```

```
catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Get Access Rights

Description

Retrieves the access rights for a specified user.

Syntax

public virtual int[] GetAccessRights(string userName)

Parameters

Name	Type	Description
userName	string	A string that contains the user name for the user. This member must be a registered AgilePoint user.

Output

An array of integers that represent the access rights of the user.

Example

```
IWFAdminService svc = GetAdminService();
string userName = ...// for example, "vit-34\\ap_svc"

try
 {
    string[] accessNames = svc.GetAccessRightNames();
    int[] userRights = svc.GetAccessRights(userName);
    Console.WriteLine("The user has the following rights:");
```

```
    foreach (int rightCode in userRights)
        {
        Console.WriteLine(accessNames[rightCode]);
        }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
/*
This example produces the following results:
The user has the following rights:
Register and modify the user information
Unregister a user
Add and modify the role information
Remove a role
Add and modify the group information
Remove a group
Modify/View the system configuration
Add a process definition
Checkin and checkout a process definition
Delete and disable a process definition
...
*/
```

Supported Versions

3.2.0.4 and higher

Get Group

Description

Retrieves a group object with the specified group name.

Syntax

public virtual WFGroup GetGroup(string groupName)

Parameters

Name	Type	Description
groupName	string	A string that contains the name of the group.

Output

WFGroup object. If the specified group does not exist, the output is null.

Example

```
IWFAdminService svc = GetAdminService();
string groupName = ...// for example, "Administrators";

try
 {
    WFGroup grpInfo = svc.GetGroup(groupName);
    Console.WriteLine("Group Name:{0}; Group Lead:{1}",
 grpInfo.Name,
    grpInfo.ResponsibleUser);
 }

catch (Exception ex)
 {
 Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
/*
This example produces the following results:
Group Name:Administrators; Group Lead:\Administrator
*/
```

Supported Versions

3.2.0.4 and higher

Get Group Members

Description

Retrieves the members of a specified group.

Syntax

public virtual WFGroupMember[] GetGroupMembers(string groupName)

Parameters

Name	Type	Description
groupName	string	A string that contains the name of the group.

Output

An array of WFGroupMember objects.

Example

```
IWFAdminService svc = GetAdminService();
string groupName = ...// for example, "Administrators";

try
 {
    WFGroupMember[] grpMembers =
 adminService.GetGroupMembers(groupName);
    Console.WriteLine("AgilePoint Group {0} has {1} members:",
    groupName, grpMembers.Length);
    foreach (WFGroupMember grpMember in grpMembers)
        {
        Console.WriteLine("Member Name:{0}",
 grpMember.Member);
        }
 }

catch (Exception ex)
 {
 Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }

/*
This example produces the following results:
AgilePoint Group Administrators has 3 members:
Member Name:\Administrator
Member Name:VITBDC\yuvarajn
Member Name:vit-34\manager
*/
```

Supported Versions

3.2.0.4 and higher

Get Groups

Description

Retrieves all the group objects in the system.

Syntax

public virtual WFGroup[] GetGroups()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

An Array of WFGroup objects.

Example

```
IWFAdminService svc = GetAdminService();

try
 {
    WFGroup[] apGroups = svc.GetGroups();
    Console.WriteLine("AgilePoint Group {0} has {1} members:",
    groupName, grpMembers.Length);
    Console.WriteLine("AgilePoint Groups:");
    foreach(WFGroup grp in apGroups)
        {
        System.Console.WriteLine("Name:{0};Group Lead:{1} ",
 grp.Name, grp.ResponsibleUser);
        }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }

/*
This example produces the following results:
AgilePoint Groups:
```

```
Name:Administrators;Group Lead:\Administrator
Name:Domain Users;Group Lead:VITBDC\amarnv
*/
```

Supported Versions

3.2.0.4 and higher

Get Role

Description

Retrieves a role object by name.

Syntax

public virtual WFRole GetRole(string roleName);

Parameters

Name	Type	Description
roleName	string	A string that contains the name of a role.

Output

WFRole object with the specified role name.

Example

```
IWFAdminService svc = GetAdminService();
string rolName = ...// for example, "Administrators"

try
 {
    WFRole role = adminService.GetRole(rolName);
    Console.WriteLine("Name = '" + role.Name + "' Description
 = '" +
    role.Description + "'");
 }

catch (Exception ex)
```

```
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }

/* This example produces the following results:
Name = 'Administrators' Description = 'Administrators have
 complete access to maintain system' */
```

Supported Versions

3.2.0.4 and higher

Get Roles

Description

Retrieves a list of all roles in the system.

Syntax

public virtual WFRole[] GetRoles()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

An array of WFRole objects that includes all roles.

Example

```
IWFAdminService svc = GetAdminService();

try
 {
    WFRole[] roles = adminService.GetRoles();
    foreach (WFRole role in roles)
        {
        Console.WriteLine("Name = '{0}', Description =
 '{1}'",role.Name,
```

```
        role.Description);
        }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }

/* This example produces the following results:
Name = 'Administrators' Description = 'Administrators have
 complete
access to maintain system'
Name = 'Process Runtime Managers' Description = 'Process
 Runtime
Managers have complete access to manage runtime processes'
Name = 'process definition Designers' Description = 'process
definition Designers have complete access to add, modify and
 remove process
templates' */
```

Supported Versions

3.2.0.4 and higher

Query Role Members

Description

Retrieves the members assigned to a role that match a specified SQL statement.

Syntax

public virtual WFRoleMember[] QueryRoleMembers(string roleName, string sql)

Parameters

Name	Type	Description
roleName	string	A string that contains the name of a role.
sql	string	A string that contains the where clause of the SQL statement you want to query.

Output

Returns array of WFRoleMember members of the role that match the specified SQL statement.

Example

```
IWFAdminService svc = GetAdminService();
string roleName = ...// for examples, "Users"
string where = ...// for example, "ASSIGNEE_TYPE='User'"

try
 {
    WFRoleMember[] roleMembers =
 svc.QueryRoleMembers(roleName, where);
    foreach (WFRoleMember member in roleMembers)
        {
        Console.WriteLine("Assignee = '{0}', Created Date =
 '{1}'",
                          member.Assignee,
                          member.CreatedDate.ToShortDateString());
        }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }

/*
This example produces the following results:
Assignee = 'DEMO\CONTROLLER' & Created Date = '9/18/2009'
Assignee = 'DEMO\CFO' & Created Date = '9/18/2009'
*/
```

Supported Versions

3.2.0.4 and higher

Remove Group

Description

Removes a group from the AgilePoint system.

Syntax

public virtual void RemoveGroup(string groupName)

Parameters

Name	Type	Description
groupName	string	A string that contains the name of the group.

Output

None.

Example

```
IWFAdminService svc = GetAdminService();
string roleName = ...// for examples, "Users"

try
 {
    svc.RemoveGroup(roleName);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Remove Group Member

Description

Removes a member from a group.

Syntax

public virtual void RemoveGroupMember(string groupName, string userName)

Parameters

Name	Type	Description
groupName	string	A string that contains the name of the group.
userName	string	A string that contains the user name for the user. This member must be a registered AgilePoint user.

Output

None.

Example

```
IWFAdminService svc = GetAdminService();
string roleName = ...// for examples, "Engineers"
string userName = ...// for example, @"VIT\ct-002"

try
 {
    svc.RemoveGroupMember(roleName, userName);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Remove Role

Description

Removes a role from the AgilePoint system.

Syntax

public virtual void RemoveRole(string roleName)

Parameters

Name	Type	Description
roleName	string	A string that contains the name of a role.

Output

None.

Example

```
IWFAdminService svc = GetAdminService();
string roleName = ...// for example, "Engineers"

try
 {
    svc.RemoveRole (roleName);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Remove Role Member

Description

Removes a user or a group from a specified role.

Syntax

public virtual void RemoveRoleMember(string roleName, string assignee, string assigneeType, string objectID)

Parameters

Name	Type	Description
roleName	string	A string that contains the name of a role.
assignee	string	The registered user name or group name.
assigneeType	string	The type for the assignee: Either user or group.
objectID	string	Reserved for future use. Leave null.

Output

None.

Example

```
IWFAdminService svc = GetAdminService();
string roleName = ...// for example, "Engineers"
string userName = ...// for example, @"VIT\ct-002"

try
 {
    svc.RemoveRoleMember(roleName, userName, "User", null);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
```

```
}
```

Supported Versions

3.2.0.4 and higher

Update Group

Description

Updates information for a group.

Syntax

WFGroup UpdateGroup(string groupName, string description, string responsibleUser, bool enabled)

Parameters

Name	Type	Description
groupName	string	A string that contains the name of the group.
description	string	A string that contains the description of an entity.
responsibleUser	string	A string that contains the name of the responsible user of this group. The responsible user must be a registered AgilePoint user.
enabled	bool	A Boolean value: True enables the entity; False disables the entity.

Output

An updated WFGroup.

Example

```
IWFAdminService svc = GetAdminService();
```

```
string groupName = ...// for example, "TestGroup1"
string description = ... // for example, "This is new
 description of
the user group"
string groupLeadUserName = ...// for example, "DEMO3\
\Administrator"

try
 {
WFGroup updatedGroup = Svc.UpdateGroup(groupName, description,
groupLeadUserName, true);
 }

catch (Exception ex)
 {
Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Update Role

Description

Updates information for a role.

Syntax

public virtual WFRole UpdateRole(string roleName, string description, int[] rights, bool enabled)

Parameters

Name	Type	Description
roleName	string	A string that contains the name of a role.
description	string	A string that contains the description of an entity.
rights	WFAccessRights	Array of index values that specifies the rights that are assigned to the Role. See the following table for appropriate

Name	Type	Description
		indexes. WFAccessRights provides the enums for rights. See the sample code for more information.
enabled	bool	A Boolean value: True enables the entity; False disables the entity.

Output

An updated WFRole object.

Example

```
IWFAdminService svc = GetAdminService();
string roleName = ...// for example, "TestRole"
string description = ...// for example, "This is new
 description for TestRole"
List<int> list = new List<int>();
list.Add(WFAccessRights.AddModifyGroup);
list.Add(WFAccessRights.AddModifyRole);
list.Add(WFAccessRights.AddModifyUser);
list.Add(WFAccessRights.AddProcessTemplate);
list.Add(WFAccessRights.AddRemoveModifyDelegation);
int[] rights = list.ToArray();

try
 {
    WFRole updatedRole = svc.UpdateRole(roleName, description
 rights, true);
 }

catch( Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

User Delegation

This section describes service calls related to user delegation processes.

Add Delegation

Description

Creates a rule for delegating one user's tasks to another user. This method uses the WFDelegation class. For more information, see the AgilePoint Class Reference.

Syntax

public virtual WFDelegation AddDelegation(WFDelegation delegation)

Parameters

Name	Type	Description
delegation	WFDelegation	An object that specifies the details of the delegation rule, including the user whose tasks will be delegated and the designated user to whom to delegate the tasks.

Output

WFDelegation object.

Example

```
IWFAdminService svc = GetAdminService();
WFDelegation delegation = new WFDelegation();

//Set the object properties.
delegation.FromUser = "Demo3\\Andy";
delegation.ToUser = "Demo3\\Joe";
delegation.StartDate = DateTime.Now;
delegation.EndDate = DateTime.Parse("27/10/2009");
delegation.Description = "Delegating Andy's task to Joe";

try
    {
    WFDelegation delegation =
 adminService.AddDelegation(delegation);
    Console.WriteLine("Delegation ID: {0}",
 delegation.DelegationID);
```

```
    }

catch (Exception ex)
    {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Activate Delegation

Description

Activates a delegation.

Syntax

public virtual void ActivateDelegation(string delegationID)

Parameters

Name	Type	Description
delegationID	string	A string that represents the unique ID of a delegation object.

Output

None.

Example

```
IWFAdminService svc = GetAdminService();
string delegationID = ...;

try
    {
    svc.ActivateDelegation(delegationID);
    }

catch (Exception ex)
```

```
    {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0 and higher

Cancel Delegation

Description

Cancels a currently operating delegation.

Syntax

public virtual void CancelDelegation(string delegationID)

Parameters

Name	Type	Description
delegationID	string	A string that represents the unique ID of a delegation object.

Output

None.

Example

```
IWFAdminService svc = GetAdminService();
string delegationID = ...;

try
    {
    svc.CancelDelegation(delegationID);
    }

catch (Exception ex)
    {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
```

```
    }
```

Supported Versions

3.2.0.4 and higher

Get Delegation

Description

Retrieves a delegation object.

Syntax

public virtual WFDelegation GetDelegation(string delegationID)

Parameters

Name	Type	Description
delegationID	string	A string that represents the unique ID of a delegation object.

Output

WFDelegation object that specifies the user whose tasks will be delegated and the designated user to whom to delegate tasks.

Example

```
IWFAdminService svc = GetAdminService();
string delegationID = ...; // for example,
 "C9A40F4BDA26481FB822C398C4387901"

try
 {
    WFDelegation delegation = svc.GetDelegation(delegationID);
    Console.WriteLine("Delegation Id:{0}; From User:{1}; To
 User:{2};
    Status:{3}",
                    delegation.DelegationID,
                    delegation.FromUser,
                    delegation.ToUser,
```

```
                    delegation.Status);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }

/* Output
Delegation Id:C9A40F4BDA26481FB822C398C4387901; From
User:VITBDC\yuvarajn; To User:VITBDC\ravis; Status:Canceled
*/
```

Supported Versions

3.2.0.4 and higher

Get Delegations

Description

Retrieves a list of delegation objects that match the specified parameters. You can leave the parameters null to indicate any.

Syntax

public virtual WFDelegation[] GetDelegations(string fromUser, string toUser, string status)

Parameters

Name	Type	Description
fromUser	string	A string value that represents the user whose tasks will be delegated.
toUser	string	The designated user who will receive the delegated tasks.
status	string	The status of the associated item.

Output

An array of WFDelegation objects.

Example

```
IWFAdminService svc = GetAdminService();
string fromUser = ...// for example, @"vitbdc\yuvarajn"
string toUser = ...// for example, null for any
string status = WFDelegation.ACTIVE;

try
 {
    WFDelegation[] delegations = svc.GetDelegations( fromUser,
 toUser,
    status);
    foreach(WFDelegation delegation in delegations)
    {
    Console.WriteLine("Delegation Id:{0}; From User:{1}; To
 User:{2};
       Status:{3}, {4}=>{5}",
                     delegation.DelegationID,
                     delegation.FromUser,
                     delegation.ToUser,
                     delegation.Status,
                     delegation.StartDate,
                     delegation.EndDate);
    }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Remove Delegation

Description

Removes a delegation from the AgilePoint system.

Syntax

public virtual void RemoveDelegation(string delegationID)

Parameters

Name	Type	Description
delegationID	string	A string that represents the unique ID of a delegation object.

Output

None.

Example

```
IWFAdminService svc = GetAdminService();
string delegationID = ...;

try
 {
    svc.RemoveDelegation(delegationID);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Update Delegation

Description

Updates a delegation object that has already been created.

Syntax

WFDelegation UpdateDelegation(WFDelegation delegation)

Parameters

Name	Type	Description
delegation	WFDelegation	An object that specifies the details of the delegation rule, including the user whose tasks will be delegated and the designated user to whom to delegate the tasks.

Output

Returns an updated instance of WFDelegation.

Example

```
IWFAdminService svc = GetAdminService();
WFDelegation delegation = new WFDelegation();
delegation.DelegationID = ...// unique ID
delegation.FromUser = ...// from user name
delegation.ToUser = ...// to user name
delegation.StartDate = ... // start date
delegation.EndDate = ...// end date

try
 {
WFDelegation updatedDelegation =
 svc.UpdateDelegation( delegation );
 }

catch (Exception ex)
 {
Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Report Configuration Methods

This section describes service calls related to reports.

Add Report Configuration

Description

Adds a report configuration to the system.

Syntax

public virtual WFReportConfigure AddReportConfigure(string reportName, string configure)

Parameters

Name	Type	Description
reportName	string	A string that contains the name of a report.
configure	string	The report configuration in XML format.

Output

WFReportConfigure object.

Example

```
IWFAdminService svc = GetAdminService();
string reportName = ...//for example, "weekly task report"
string configure = ...// xml-serialization of
 WFReportConfiguration

try
    {
    WFReportConfigurereportConfig =
    svc.AddReportConfigure(reportName, configure);
    }

catch (Exception ex)
    {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
    }
```

Supported Versions

3.2.0.4 and higher

Get All Report Configurations

Description

Retrieves all report configurations from the system.

Syntax

public virtual WFReportConfigure[] GetAllReportConfigure()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

WFReportConfigure object.

Example

```
IWFAdminService svc = GetAdminService();

try
 {
    WFReportConfigure[] reportConfigs =
 svc.GetAllReportConfigure();
    Console.WriteLine("This AgilePoint Server has {0} reports
    configured:", reportConfigs.Length);
    foreach (WFReportConfigure config in reportConfigs)
    {
       Console.WriteLine("Report Name: {0}",
 config.ReportName);
    }
 }

catch (Exception ex)
 {
```

```
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
/*
This example produces the following results:
This AgilePoint Server has 1 reports configured:
Report Name:Average Process Time By Automatic Activities
*/
```

Supported Versions

3.2.0.4 and higher

Get Report Configuration

Description

Retrieves a report configuration from the system.

Syntax

public virtual WFReportConfigure GetReportConfigure(string reportName)

Parameters

Name	Type	Description
reportName	string	A string that contains the name of a report.

Output

WFReportConfigure object.

Example

```
IWFAdminService svc = GetAdminService();
string reportName = ...;

try
 {
    WFReportConfigure cfg =
 svc.GetReportConfigure(reportName);
```

```
    Console.WriteLine("Report Name: {0}, config:{1}",
 cfg.ReportName,
    cfg.Configure);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Remove Report Configure

Description

Removes a report configuration from the system.

Syntax

public virtual void RemoveReportConfigure(string reportName)

Parameters

Name	Type	Description
reportName	string	A string that contains the name of a report.

Output

None.

Example

```
IWFAdminService svc = GetAdminService();
string reportName = ...;

try
 {
    svc.RemoveReportConfigure(reportName);
```

```
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Update Report Configuration

Description

Updates a report configuration in the AgilePoint system.

Syntax

public virtual WFReportConfigure UpdateReportConfigure(string reportName, string configure)

Parameters

Name	Type	Description
reportName	string	A string that contains the name of a report.
configure	string	The report configuration in XML format.

Output

WFReportConfigure object.

Example

```
IWFAdminService svc = GetAdminService();
string reportName = ...//for example, "weekly task report"
string configure = ...// xml-serialization of
 WFReportConfiguration

try
```

```
 {
    WFReportConfigurereportConfig =
    svc.UpdateReportConfigure(reportName, configure);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Organization Properties

This section describes service calls related to various organization properties.

Get Organization Properties

Description

Retrieves organization properties such as Title, Department and Location.

Syntax

public virtual KeyValue[] GetOrganizationProperties(string name)

Parameters

Name	Type	Description
name	string	A string that contains the name of an item, such as a property or attribute.

Output

An array of KeyValue objects.

Example

```
IWFAdminService svc = GetAdminService();
string name = ...// for example, "Departments"

try
 {
    KeyValue[] organizationProperties =
    svc.GetOrganizationProperties(name);
    foreach (KeyValue property in organizationProperties)
    {
       Console.WriteLine("Property Name = 'Department'
 Property Value =
       '" + property.Value + "'");
    }
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Remove Organization Properties

Description

Removes an organization property from AgilePoint.

Syntax

public virtual void RemoveOrganizationProperties(string name)

Parameters

Name	Type	Description
name	string	A string that contains the name of an item, such as a property or attribute.

Output

None.

Example

```
IWFAdminService svc = GetAdminService();
string name = ...// for example, "Departments"

try
 {
    svc. RemoveOrganizationProperties (name);
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Update Organization Properties

Description

Updates organization properties in the AgilePoint the system.

Syntax

void UpdateOrganizationProperties(string name, KeyValue[] list)

Parameters

Name	Type	Description
name	string	A string that contains the name of an item, such as a property or attribute.

Name	Type	Description
list	KeyValue	A list of key-value pairs you want to add to the list of properties.

Output

None.

Example

```
IWFAdminService svc = GetAdminService();
string name = ...// for example, "Titles"
List<KeyValue> list = new List<KeyValue>();
list.Add( new KeyValue( "MANAGER", "Manager" ) );
list.Add( new KeyValue( "REG_SALES_DIRECTOR", "Regional Sales
Director" ) );
list.Add( new KeyValue( "GENERAL_MANAGER", "General
 Manager" ) );
KeyValue[] properties = list.ToArray();

try
 {
 adm.UpdateOrganizationProperties("Titles",properties );
 }

catch (Exception ex)
 {
Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Component Administration Methods

This section describes service calls related to server component administration.

Get Server Component

Description

Retrieves a server component name.

Syntax

public virtual WFComponent GetServerComponent(string name)

Parameters

Name	Type	Description
name	string	A string that contains the name of an item, such as a property or attribute.

Output

WFComponent object.

Example

```
None.
```

Supported Versions

3.2.0.4 and higher

Get Server Component Names

Description

Retrieves the server component names using the Admin Services.

Syntax

public virtual string[] GetServerComponentNames()

Parameters

Name	Type	Description
None	Not applicable	Not applicable

Output

An array of strings that contain the server component names.

Example

```
IWFAdminService svc = GetAdminService();

try
 {
    string[] names = svc.GetServerComponentNames();
 }

catch (Exception ex)
 {
    Console.WriteLine("Failed! " + ShUtil.GetSoapMessage(ex));
 }
```

Supported Versions

3.2.0.4 and higher

Classes

This section includes references for all classes within the AgilePoint Web Service API.

IWFWorkflowService

Description

A class that provides interfaces for the AgilePoint workflow API on the client side.

Syntax

public interface `IWFWorkflowService`

Constructors

Not Applicable.

Namespace and Assembly

Requirement	Value
Namespace	Ascentn.Workflow.Base
Assembly	Ascentn.Workflow.WFBase (in Ascentn.Workflow.WFBase.dll)

KeyValue

Description

A class that represents an object with name and value properties.

Syntax

public class `KeyValue`

Constructors

public ();

public (string key, string val);

Namespace and Assembly

Requirement	Value
Namespace	Ascentn.Workflow.Base
Assembly	Ascentn.Workflow.WFBase (in Ascentn.Workflow.WFBase.dll)

Properties

Property	Type	Description
Key	string	Gets and sets the key of this instance.
Value	string	The value for an item, such as the value for an attribute in a name-value pair.

NameValue

Description

A class that represents an object with the name and value properties.

Syntax

public class `NameValue`

Constructors

public `NameValue`();

public `NameValue`(string *name*,object *value*);

Namespace and Assembly

Requirement	Value
Namespace	Ascentn.Workflow.Base
Assembly	Ascentn.Workflow.Share (in Ascentn.Workflow.Share.dll)

Properties

Property	Type	Description
Name	string	The name of an item, such as the name of an attribute in a name-value pair.
Value	object	The value for an item, such as the value for an attribute in a key-value pair.

RegisteredUser

Description

A class that represents an AgilePoint registered user.

Syntax

public class `RegisteredUser`

Constructors

public ();

public (string user, string emailAddress, DateTime registeredDate, string fullName);

Namespace and Assembly

Requirement	Value
Namespace	Ascentn.Workflow.Base
Assembly	Ascentn.Workflow.WFBase (in Ascentn.Workflow.WFBase.dll)

Properties

Property	Type	Description
Department	string	Gets and sets department of this user
EMailAddress	string	Gets and sets e-mail address of this user
FullName	string	Gets and sets full name of this user
Manager	string	Gets and sets manager of this user
RegisteredDate	System.DateTime	Gets and sets date registered
Title	string	Gets and sets title of this user
UserName	string	Gets and sets qualified user name of this instance qualified user name formats as [Domain Name]\[Logon Username] or [Local host name]\[Logon Username]

WFAccessRights

Description

A class that represents different types of access rights.

Syntax

public enum `WFAccessRights`

Constructors

Not Applicable.

Namespace and Assembly

Requirement	Value
Namespace	Ascentn.Workflow.Base
Assembly	Ascentn.Workflow.WFBase (in Ascentn.Workflow.WFBase.dll)

WFAny

Description

A class that represents a primitive data type with type code.

Syntax

public class `WFAny`

Constructors

public ();

Namespace and Assembly

Requirement	Value
Namespace	Ascentn.Workflow.Base
Assembly	Ascentn.Workflow.WFBase (in Ascentn.Workflow.WFBase.dll)

Properties

Property	Type	Description
Type	int	Gets and sets the type of a WFAny object.
Value	object	The value for an item, such as the value for an attribute in a key-value pair.

WFEvent

Description

A class that represents a workflow event.

Syntax

public class `WFEvent`

Constructors

public ();

public (string name);

public (string sender, string name);

Namespace and Assembly

Requirement	Value
Namespace	Ascentn.Workflow.Base
Assembly	Ascentn.Workflow.WFBase (in Ascentn.Workflow.WFBase.dll)

Properties

Property	Type	Description
ActivityInstID	string	Gets and sets activity instance ID.
AutoStart	bool	Gets and sets the value that determines if the process should start immediately after it is created.
ClientData	string	Gets and sets client data.
CustomAttributes	NameValue	Gets and sets custom attributes in XML format
Designated	bool	Gets and sets the flag that determines if the event should be handled by a designated engine.
Diagnostic	bool	Gets and sets the flag that determines if the process data will be deleted from database after the process is completed.
EndDate	System.DateTime	Gets and sets the completion date of event.
Entries	int	Gets the relay time of the event.
Error	string	Gets and sets the error message of the event.
EventID	string	Gets and set the event ID.
EventName	string	Gets and sets the name of the event.
ParamsXml	string	Gets and sets the parameters as XML.
ParentProcInstID	string	Gets and sets parent process instance ID.
ProcDefID	string	Gets and sets process template (process definition) ID.
ProcInstID	string	Gets and sets process instance ID.

Property	Type	Description
ProcInstName	string	Gets and sets process instance name.
Sender	string	Gets and sets sender of the event.
SentDate	System.DateTime	Gets and sets the sent date of event.
SourceWorkItemID	string	Gets and sets work item (task) ID.
Status	string	Gets and sets the status of the event.
ThrowAwayInstance	bool	Gets and sets the flag that determines if AgilePoint server should clear the process from cache
UserID	string	Gets and sets user ID.
WorkItemID	string	Gets and sets work item(task) ID.
WorkObjectID	string	Gets and sets work object ID.

WFPartialRollbackInstruction

Description

A class that provides instructions for activating activity instances.

Syntax

public class `WFPartialRollbackInstruction`

Constructors

public ();

Namespace and Assembly

Requirement	Value
Namespace	Ascentn.Workflow.Base
Assembly	Ascentn.Workflow.WFBase (in Ascentn.Workflow.WFBase.dll)

Properties

Property	Type	Description
PartialRollbackUnits	PartialRollbackUnit[]	An array of instances of the PartialRollbackUnit class.

WFProcessMergingInstruction

Description

A class that provides instructions for merging process instances during runtime.

Syntax

public class WFProcessMergingInstruction

Constructors

Not Applicable.

Namespace and Assembly

Requirement	Value
Namespace	Ascentn.Workflow.Base
Assembly	Ascentn.Workflow.WFBase (in Ascentn.Workflow.WFBase.dll)

Properties

Property	Type	Description
MergingProcessInstanceIDs	string	An array that includes the process instance IDs of the process instances you are merging.
MergedProcessInstance	MergedProcessParameter	An instance of the MergedProcess Parameter class that includes the merged process instance.

WFProcessMigrationInstruction

Description

A class that provides instructions for merging process instances during runtime.

Syntax

public class `WFProcessMigrationInstruction`

Constructors

Not Applicable.

Namespace and Assembly

Requirement	Value
Namespace	Ascentn.Workflow.Base
Assembly	Ascentn.Workflow.WFBase (in Ascentn.Workflow.WFBase.dll)

Properties

Property	Type	Description
MatchingActivityDefinition	class	Provides information for activity migration.
SourceProcessDefinitionID	string	The ID of the original, or source, process definition.
TargetProcessDefinitionID	string	The ID of the target, or destination, process definition.
Action	MigrationAction	The migration action.
IncludeXmlData	bool	Specifies whether a migration includes XML data.

WFProcessSplittingInstruction

Description

A class that provides instructions for splitting a process instance during runtime.

Syntax

public class `WFProcessSplittingInstruction`

Constructors

Not Applicable.

Namespace and Assembly

Requirement	Value
Namespace	Ascentn.Workflow.Base
Assembly	Ascentn.Workflow.WFBase (in Ascentn.Workflow.WFBase.dll)

Properties

Property	Type	Description
SplitProcessInstances	SplitProcessParameter[]	An array of instances of the SplitProcessParameter class.

WFQueryExpr

Description

A class that represents a query expression object.

Syntax

public class `WFQueryExpr`

Constructors

public ();

public (string columnName, int op, WFAny any, bool val);

Namespace and Assembly

Requirement	Value
Namespace	Ascentn.Workflow.Base
Assembly	Ascentn.Workflow.WFBase (in Ascentn.Workflow.WFBase.dll)

Properties

Property	Type	Description
Any	WFAny	Gets and sets the WFAny to be compared.

Property	Type	Description
ColumnName	string	Gets and sets the associated column name in database.
IsValue	bool	Gets and sets the flag that indicates if it compares a value or column of table
Name	string	The name of an item, such as the name of an attribute in a name-value pair.
Operator	int	Gets and sets the operator used for comparison.

WFTimeDuration

Description

A class that represents Time Duration with length, time unit, and business time.

Syntax

public class `WFTimeDuration` : Serializable

Constructors

public ();

public (string length, WFTimeUnit unit, bool b);

public (int length, WFTimeUnit unit, bool b);

Namespace and Assembly

Requirement	Value
Namespace	Ascentn.Workflow.Base
Assembly	Ascentn.Workflow.WFBase (in Ascentn.Workflow.WFBase.dll)

Properties

Property	Type	Description
BusinessTime	bool	Determines whether the system should calculate duration as business time
Length	string	Length of time duration
Unit	WFTimeUnit	Time unit, hour, day, week, month, and year

WFTimeUnit

Description

A class that represents time unit.

Syntax

public class `WFTimeUnit`

Constructors

public ();

Namespace and Assembly

Requirement	Value
Namespace	Ascentn.Workflow.Base
Assembly	Ascentn.Workflow.WFBase (in Ascentn.Workflow.WFBase.dll)

Properties

Property	Type	Description
Value	int	The value for an item expressed as an integer, such as a time unit.

Parameters

A list of all parameters used in the AgilePoint Web Service API

Name	Type	Description
activityInstanceID	string	A string that contains the ID for an activity instance.
_api	IWFWorkflowService	A null parameter.
appName	string	A string that contains the name of the application. The name is case-sensitive.
assignee	string	The registered user name or group name.
assigneeType	string	The type for the assignee: Either user or group.
attachments	string	File attachments included with the email. This parameter must use a file path from the file system (for example, C:\file.txt) on the machine where AgilePoint Server is installed. If there is no attachment, you can pass **null** or **String.Empty**.
attribute	NameValue	A NameValue array that contains the attributes that needs to be updated in the work item. For a manual work item, the following attributes can be updated: • NAME • ORIGINAL_USER_ID • CLIENT_DATA, POOL_ID • POOL_INFO • STATUS • USER_ID • PRIORITY

Name	Type	Description
		• DUE_DATE For an automatic work item, the following attributes can be updated: • DUE_DATE • STATUS - if the value is Canceled, Completed, Overdue, Running, or Waiting. For an automatic work item, only DUE_DATE can be updated.
attributeName	string	A string that contains the name of the process instance attribute you want to retrieve.
attributes	NameValue	Name-value pairs associated with a custom ID.
basePID	string	The ID of the base process definition.
bDependent	bool	If true: The source work item waits until the linked work item is either completed or canceled, before it can be completed/canceled. If false: The source work item can be completed/canceled regardless of whether the linked work item is completed/canceled.
body	string	The text of the email (body).
cc	string	Persons copied on the email.
clientData	string	A string that contains the client data. If clientData is null, the system will keep existing client data, otherwise the relevant data would be overwritten.
configure	string	The report configuration in XML format.
customID	string	A work object ID as identified in the process instance. This method allows only one custom ID (work object ID).

Name	Type	Description
customIDs	string	An array of strings that contains custom IDs (work object IDs).
delegation	WFDelegation	An object that specifies the details of the delegation rule, including the user whose tasks will be delegated and the designated user to whom to delegate the tasks.
delegationID	string	A string that represents the unique ID of a delegation object.
description	string	A string that contains the description of an entity.
duration	WFTimeDuration	A time-duration object that specifies the duration setting of the work item.
enabled	bool	A Boolean value: True enables the entity; False disables the entity.
eventID	string	A unique, 32-character event ID.
expr	WFQueryExpr	An object that represents the where clause of a SQL query expression.
filter	string	A string that contains the wildcard-enabled group filter in the format (name=<<wildcard filter>>)
from	string	The sender of the email.
fromUser	string	A string value that represents the user whose tasks will be delegated.
groupDistinguishedName	string	A string that contains the name of the group in LDAP format.
groupName	string	A string that contains the name of the group.
initiator	string	A string that contains the user name of the user who initiates process.

Name	Type	Description
instruction	WFPartialRollbackInstruction	An object that specifies the instructions for the partial rollback.
instruction	WFProcessMergingInstruction	An object that specifies the instructions for the merge.
instruction	WFProcessMigrationInstruction	An object that specifies the instructions for the migration.
instruction	WFProcessSplitting	An object that specifies the instructions for splitting the process instance.
list	KeyValue	A list of key-value pairs you want to add to the list of properties.
locale	string	A string that contains the client locale in the format en-US.
mID	string	A string that contains the unique ID for an email notification.
name	string	A string that contains the name of an item, such as a property or attribute.
names	string	An array of strings that contains the names of items, such as a properties or attributes.
newAssigneeUserID	string	The User ID for the user to which you want to assign the work item.
objectID	string	Reserved for future use. Leave null.
originalUserID	string	The user ID for the original user assigned the work item.
PID	string	The process definition ID for a released process definition.
PIID	string	A 32 character unique process instance ID for the process instance you are creating. If you set this value to null, the AgilePoint Server generates the ID.
PIName	string	A unique process name that is associated with the process

Name	Type	Description
		definition. The maximum length of process instance name is 1024 characters.
priority	enum	The email's priority. Values can be **high**, **normal**, or **low**.
processDefinitionName	string	The name of the process definition.
processInstanceID	string	A string that contains the ID of the process instance.
processTemplateID	string	The unique identifier for the process definition to be checked out for modification.
reserved	string	null.
responsibleUser	string	A string that contains the name of the responsible user of this group. The responsible user must be a registered AgilePoint user.
sql	string	A string that contains the where clause of the SQL statement you want to query.
startImmediately	bool	An obsolete, legacy parameter that must be true.
subject	string	The subject of the email.
superPIID	string	A 32-character unique process instance ID that acts as a parent process instance of the process instance that is intended to create. In other words, this is the ID of the process instance on which you want to base your new process instance.
to	string	Recipient of the email.
toUser	string	The designated user who will receive the delegated tasks.
user	RegisteredUser	A RegisteredUser object that contains user information.
reportName	string	A string that contains the name of a report.

Name	Type	Description
rights	WFAccessRights	Array of index values that specifies the rights that are assigned to the Role. See the following table for appropriate indexes. WFAccessRights provides the enums for rights. See the sample code for more information.
roleName	string	A string that contains the name of a role.
source	string	A string that contains the LDAP path to the domain. If the value is null, the AgilePoint Server machine domain will be used.
sourceWorkItemID	string	A unique, 32-character ID for the original, or source, work item.
status	string	The status of the associated item.
userIcon	byte	Byte array representation of icon image.
userID	string	A string that contains a user ID for the user associated with the work item.
userName	string	A string that contains the user name for the user. This member must be a registered AgilePoint user.
val	object	An object that contains the value of the custom attribute.
wID	string	A 32-byte unique work item (task) ID that represents a manual work item.
workObjectID	string	A 256-character ID for an object, such as a document, that is associated with the process instance. (Even though the field size is 256 characters, in common practice, this will usually return a 32-character GUID.)

Name	Type	Description
workObjectInfo	string	A 1024-character string associated with the process instance. Usually this parameter is used to hold more information about the work object, such as a URL for a document, within the process instance.
workToPerform	string	A string that specifies the work to perform.
xml	string	A string that contains process definition in XML format. To generate the process definition file in XML format, in AgilePoint Envision, click **File > Export & Import > Save As Deploying File(xml)**. You can also download the process definition XML from AgilePoint Enterprise Manager.

Data Types

A list of all data types used in the AgilePoint Web Service API. Common types refer to the documentation on msdn.com

Name	Description
bool	See the documentation on MSDN.
byte	See the documentation on MSDN.
DateTime	See the documentation on MSDN.
enum	See the documentation on MSDN.
int	See the documentation on MSDN.
IWFWorkflowService	Instantiates the IWFWorkflowService class.
KeyValue	Instantiates the KeyValue class.
MergedProcessParameter	Instantiates the MergedProcessParameter class.
MigrationAction	Instantiates the MigrationAction class.
NameValue	Instantiates the NameValue class.
object	See the documentation on MSDN.
PartialRollbackUnit	Instantiates the PartialRollbackUnit class.
RegisteredUser	Instantiates the RegisteredUser class.
SplitProcessParameter	Instantiates the SplitProcessParameter class.
String	See the documentation on MSDN.
WFAccessRights	Instantiates the WFAccessRights class.
WFAny	Instantiates the WFAny class.
WFEvent	Instantiates the WFEvent class.
WFQueryExpr	Instantiates the WFQueryExpr class.
WFTimeDuration	Instantiates the WFTimeDuration class.
WFTimeUnit	See the documentation on MSDN.

Class Properties

A list of all properties within the Web Service API Classes

Property Name	Type	Description
Action	MigrationAction	The migration action.
ActivityInstID	string	Gets and sets activity instance ID.
Any	WFAny	Gets and sets the WFAny to be compared.
AutoStart	bool	Gets and sets the value that determines if the process should start immediately after it is created.
BusinessTime	bool	Determines whether the system should calculate duration as business time
ClientData	string	Gets and sets client data.
ColumnName	string	Gets and sets the associated column name in database.
CustomAttributes	NameValue	Gets and sets custom attributes in XML format
customAttributes	NameValue	Gets and sets custom attributes in XML format
Department	string	Gets and sets department of this user
Designated	bool	Gets and sets the flag that determines if the event should be handled by a designated engine.
Diagnostic	bool	Gets and sets the flag that determines if the process data will be deleted from database after the process is completed.
EMailAddress	string	Gets and sets e-mail address of this user
EndDate	System.DateTime	Gets and sets the completion date of event.

Property Name	Type	Description
Entries	int	Gets the relay time of the event.
Error	string	Gets and sets the error message of the event.
EventID	string	Gets and set the event ID.
EventName	string	Gets and sets the name of the event.
FullName	string	Gets and sets full name of this user
IncludeXmlData	bool	Specifies whether a migration includes XML data.
IsValue	bool	Gets and sets the flag that indicates if it compares a value or column of table
Key	string	Gets and sets the key of this instance.
Length	string	Length of time duration
Manager	string	Gets and sets manager of this user
MatchingActivityDefinition	class	Provides information for activity migration.
MergedProcessInstance	MergedProcessParameter	An instance of the MergedProcess Parameter class that includes the merged process instance.
MergingProcessInstanceIDs	string	An array that includes the process instance IDs of the process instances you are merging.
Name	string	The name of an item, such as the name of an attribute in a name-value pair.
Operator	int	Gets and sets the operator used for comparison.
ParamsXml	string	Gets and sets the parameters as XML.
ParentProcInstID	string	Gets and sets parent process instance ID.

Property Name	Type	Description
PartialRollbackUnits	PartialRollbackUnit[]	An array of instances of the PartialRollbackUnit class.
ProcDefID	string	Gets and sets process template (process definition) ID.
ProcessInstanceID	string	Gets and sets process instance ID.
ProcessInstanceName	string	Gets and sets process instance name.
ProcInstID	string	Gets and sets process instance ID.
ProcInstName	string	Gets and sets process instance name.
RegisteredDate	System.DateTime	Gets and sets date registered
Sender	string	Gets and sets sender of the event.
SentDate	System.DateTime	Gets and sets the sent date of event.
SourceProcessDefinitionID	string	The ID of the original, or source, process definition.
SourceWorkItemID	string	Gets and sets work item (task) ID.
SplitProcessInstances	SplitProcessParameter[]	An array of instances of the SplitProcessParameter class.
Status	string	Gets and sets the status of the event.
TargetProcessDefinitionID	string	The ID of the target, or destination, process definition.
ThrowAwayInstance	bool	Gets and sets the flag that determines if AgilePoint server should clear the process from cache
Title	string	Gets and sets title of this user
Type	int	Gets and sets the type of a WFAny object.
Unit	WFTimeUnit	Time unit, hour, day, week, month, and year
UserID	string	Gets and sets user ID.

Property Name	Type	Description
UserName	string	Gets and sets qualified user name of this instance qualified user name formats as [Domain Name]\[Logon Username] or [Local host name]\[Logon Username]
Value	int	The value for an item expressed as an integer, such as a time unit.
Value	object	The value for an item, such as the value for an attribute in a key-value pair.
Value	string	The value for an item, such as the value for an attribute in a name-value pair.
WorkItemID	string	Gets and sets work item(task) ID.
WorkObjectID	string	Gets and sets work object ID.
WorkObjectInfo	string	Gets and sets information about a work object.