Web Services

DELMIA Apriso 2021 Implementation Guide

3DEXPERIENCE®

DASSAULT SYSTEMES | The 3DEXPERIENCE® Company

Contents

Figures

1 Introduction

1.1 Abstract

The purpose of this document is to provide an overview of the Web Services functionality within DELMIA Apriso while using a sample demonstration scenario as a reference. The mechanism built into DELMIA Apriso allows for publishing Standard Operations as Web Services and executing remote Web Services from Process Builder.

- The scope of this guide covers the following areas:
- Publishing Standard Operations as Web Services
- Configuring the security of Web Services
- Consuming external Web Services from Process Builder
- DELMIA Apriso to DELMIA Apriso communication

This guide is intended for individuals responsible for performing the system implementation that have an understanding of the Operations created in Process Builder as well as have a basic knowledge of the Windows system and Internet Information Services configuration.

1.2 Overview

Publishing Operations as Web Services

The Web Services published by DELMIA Apriso can be configured using the Process Builder Web Services Manager that modifies the configuration data stored in the database and in files in a Web server folder. Advanced configuration of Web Services can be performed on the IIS administration screen and manually in configuration files.

Web Services exposing Operations are hosted in IIS as WCF (Windows Communication Foundation) services. The use of WCF covers all of the areas related to transport, security, encoding, transactional processing, metadata exchange, and other aspects related to Web Services. DELMIA Apriso benefits from all of the flexibility, configuration, and monitoring features built into IIS.

Invoking Remote Web Services from DELMIA Apriso

DELMIA Apriso is also able to consume external Web Services. External Web Services can be called from Operations and Processes using a special Function type named Web Service. This Function requires the address of the WSDL document of an external Web Service.

Communication between DELMIA Apriso servers can be realized within Process Builder by a feature of a Sub Operation Function that allows for specifying the name of the DELMIA Apriso server on which the Operation will be executed. The Operation is called via the Web Service exposed on the remote server.

1.3 Architecture

1.3.1 Execution of an Operation Published as a Web Service

When an external application calls DELMIA Apriso Web Service, the following processing takes place:

1. The security configuration is verified by WCF based on the settings in the configuration files. This verification does not involve DELMIA Apriso and can include features such as certification comparison, digital signatures verification, or decryption of the message.
2. The Web Service described by the SVC file starts to execute.
3. The system checks the DELMIA Apriso security by validating the provided login name, password, and Role membership required to invoke the Operation. If the security validation fails, an error message is returned to the caller.
4. The system determines the specified or dynamic revision of a published Operation.
5. The system decodes the parameters passed as a structure to the Web Service, and then converts them into a format acceptable by Function Interpreter (e.g., Property Bag).
6. Job Executor (JE) is invoked to perform the actual execution of the Operation.
    a. For the synchronous invocation of the Operation (when the Output values are required), JE is called by the Web Service code (IIS contacts JE via WCF) with a request to immediately execute an "ad-hoc" job. JE creates an additional processing thread and executes the job using FI before the other jobs that are waiting in the queues in the database.
    b. When the Operation is invoked asynchronously, then the Web Service code creates a regular asynchronous job in the JE queue. The Web Service execution ends after creating the job and the results of the Operation execution are not available for the Web Service caller. JE processes the job according to the priority, together with other standard jobs.

    The actual Operation execution is done inside the Job Executor service. This allows for reusing the additional features of Job Executor, such as automatic retry, execution logging, and UI for monitoring, and also provides unified background execution tracking as in other DELMIA Apriso modules.
7. When the synchronous Operation execution is finished, the result is returned by JE to IIS. It is then reformatted into a structure exposed by Web Service and returned to the caller.

1.3.2 Invoke a Web Service from Process Builder

To invoke a remote Web Service from Operations and Processes, a Function of the Web Service type must be used.

Based on the URL of the WSDL document provided by the Process author, the system will list all of the available services. The user selects the particular Web Service and the method. The Inputs and Outputs of the PB Function are automatically created. It is also possible to override the runtime URL of the Web Service using the appropriate Function Inputs.

In Process Builder, only simple types of Web Service parameters are supported. Methods using complex data types are listed as "unsupported" in the Web Service Function properties.

1.3.3 DELMIA Apriso to DELMIA Apriso Communication

The Sub Operation Function in Process Builder has the ability to call Operations from remote DELMIA Apriso servers. The remote Operation can be called in the synchronous or asynchronous mode. In the synchronous mode, Function Interpreter contacts the remote Web Service – that executes the Operation as an immediate Job Executor task – and waits for the results. In the asynchronous mode, Function Interpreter sends a job to a local Job Executor that tries to invoke the above-mentioned remote Web Service. The Web Service on the remote machine must have the proper security setting applied to allow incoming connections.

1.4 Vocabulary

WS – Web Service

URL – Uniform Resource Locator

IIS – Internet Information Services

SOA – Service Oriented Architecture

WSDL – Web Service Definition Language

WCF – Windows Communication Foundation

JE – Job Executor

FI – Function Interpreter

2 Configuration of Published Web Services

2.1 Prerequisites

The framework for Web Services is included in the standard DELMIA Apriso configuration. The proper deployment of DELMIA Apriso and the IIS Web server is required for using DELMIA Apriso Web Services.

The prerequisite for publishing Web Services using WS Manager is the proper installation of Process Builder. For more information, refer to the Process Builder Help.

2.2 Configuration Overview

The Web Services Manager embedded in Process Builder allows for configuring most parameters of published Web Services. The advanced configuration can be performed on the IIS administration screen and manually in the configuration files.

The configuration data is stored in both the database and files located on the IIS Web server. Each Operation is published as a separate WCF Web Service described by an individual configuration file.

A detailed description of the database structure, Web Services IIS folders, configuration files, and Central Configuration options is covered in subsequent sections.

2.3 Central Configuration

There are two keys in DELMIA Apriso Central Configuration that affect the published Web Services:

- "WebServicesPath" key defines the IIS folder which stores the Web Services configuration.
- "WebServicesURL" key indicates the prefix of the URL location under which the Web Services are visible by the clients.

For detailed information on configuring these keys, refer to the “Framework” section of the Central Configuration Documentation.

2.4 Database Model

The tables used to store configuration data for published Web Services are WEBSERVICE and WEBSERVICE_ROLE. These tables contain, respectively, the list of configured Web Services together with the parameters of the Standard Operations exposed by them, and the list of Roles allowed to call a certain Web Service.

WEBSERVICE Table

This table is required to manage the published Standard Operations as Web Services.

Column	Type	Description
ID	int	The identity.
Name	nvarchar(50)	The Web Service name.
FUID	nvarchar(36) – not null	Used for GPM support.
Enabled	bit	Web Service enabled flag.
Anonymous	bit	Anonymous access flag.
ObjectType	smallint	1 – Standard Operation 2 – Business Component 3 – MI Script Only Standard Operations are supported.
Parameters	nvarchar(max)	Serialized parameters, dependent on ObjectType.
OperationID	int	Standard Operation ID.
OperationCode	nvarchar(80)	Standard Operation code.
Status	smallint – not null	For future use.

WEBSERVICE_ROLE Table

This table is required to support controlling access to Web Service using Roles.

Column	Type	Description
WebServiceID	int – not null	The link to the WEBSERVICE table.
RoleID	int – not null	The link to the ROLE table.

2.5 Configuration Files

The configuration files of the published Web Services are stored in IIS Web server folders. The main folder is defined by the "WebServicesPath" entry in Central Configuration, and it contains subfolders related to the security profiles. Each subfolder has different security settings (e.g., public usage or security for AJAX clients).

Each Operation is published as a separate WCF Web Service and described by an individual SVC file. The file contains the definition of all methods required to invoke the Operation, the definition of the structures keeping the Input and Output parameters, and the structure returned as a result of the Web Service execution. This information creates a contract, which defines the external interface of a given Operation.

The following listing presents a part of the configuration file for the `ws_example_1` Web Service. The `ws_example_1.svc` configuration file is located by default in the "Public" profile subfolder in the DELMIA Apriso Web Services path. The file publishes a Standard Operation with two Inputs (of the Char and Integer types) and two Outputs (of the Boolean and DateTime types).

```
[...]
namespace FlexNet.WebServices
{
    public class ws_example_1Inputs
    {
        public System.String Input1;
        public System.Int32 Input2;
    }
    [ServiceContract(Namespace = "http://www.apriso.com/DELMIAApriso")]
    [...]
    public class ws_example_1
    {
        [...]
        [OperationContract]
        public OperationInvocationResult Invoke(ws_example_1Inputs inputs)
        {
            if(!this._enabled)
                return OperationInvocationResult.Failure("Service temporary unavailable.");
            this.ValidateInputs(inputs);
            [...]
            try
            {
                Employee employee = this.AuthenticateEmployee(inputs);
                ImmediateJob job = this.CreateJob(employee, inputs);
                ExecutionArgs executionArgs;
                Outcome result = job.Execute(out executionArgs);
                [...]
                OperationInvocationResult invocationResult =
                    OperationInvocationResult.Success();
                [...]
                invocationResult.Output_Output = (System.Boolean)
                    TypeConvert.ChangeType (outputsBag["Output1"],
                    typeof(System.Boolean));
                [...]
                invocationResult.Output_Output1 = (System.DateTime)
                    TypeConvert.ChangeType(outputsBag["Output2"],
                    typeof(System.DateTime));
                [...]
                return invocationResult;
            }
            catch (Exception e)
            {
                [...]
            }
        }
        [OperationContract]
        public OperationInvocationResult Invoke_Async(ws_example_1Inputs inputs)
        {
            if(!this._enabled)
                return OperationInvocationResult.Failure("Service temporary unavailable.");
            this.ValidateInputs(inputs);
            [...]
            try
            {
                Employee employee = this.AuthenticateEmployee(inputs);
                ImmediateJob job = this.CreateJob(employee, inputs);
                job.Save();
                return OperationInvocationResult.Success();
            }
            catch (Exception e)
            {
                [...]
            }
        }
        [...]
    }

    public class OperationInvocationResult
    {
        public bool IsSuccess;
        public string Error;
        public System.Boolean Output_Output1;
        public System.DateTime Output_Output2;
        [...]
    }
}
```

The WSDL document describing the above Web Service is visible by the clients at the following URL:

`http://<server name>/Apriso/WebServices/Public/ws_example_1.svc?wsdl`

The published Web Service contains two methods – Invoke and Invoke_Async – which allow for invoking the Operation in the synchronous or asynchronous mode.

2.6 Configuration of Security Profiles

The main folder for the Web Services defined by the "WebServicesPath" entry in Central Configuration contains subfolders related to security profiles. Each profile can contain a number of published Operations.

There are two preconfigured security profiles in the default installation:

- Public – a general purpose profile with public access (compatible with ASMX Web Services)
- AJAX – for access from AJAX client scripts

To add a new profile, create a subfolder in the location defined by "WebServicePath" (the default path is `<drive>\Program Files\Dassault Systemes\DELMIA Apriso 2021\WebSite\WebServices`.) Then place a `web.config` file appropriately configured according to the WCF documentation. The IIS directory security settings can also be used to protect the profiles.

Examples of the `web.config` files can be found in the profiles installed with DELMIA Apriso as well as in the WCF documentation.

To enable access to Web Services with the use of HTTPS, modify the security settings in each `wsHttpBinding` and `webHttpBinding` defined in the `web.config` file for the given service. Changing the value of the `<security>` key to:

```
<security mode="Transport">
    <transport clientCredentialType="None" />
</security>
```

Add the above value to the binding, in case the <security> key is not present in the binding configuration.

For more information on configuring HTTPS communication, refer to the Security Implementation Guide.

Independent of the security profiles, each published Web Service can have DELMIA Apriso authentication enabled, which is based on the DELMIA Apriso login name, password, and assigned Roles.

2.7 Configuring AJAX Script to Invoke an Operation

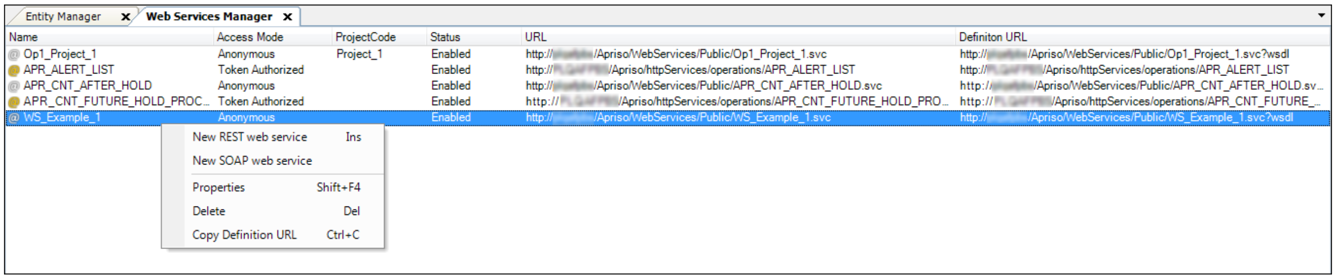
The following listing presents an example script named **InvokeOperation.aspx** that invokes an Operation which requires two Input parameters (Input1 and Input2) and returns two Output parameters (Output1 and Output2). The Operation is published as a Web Service in the AJAX profile under the name "ajaxtest."

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="InvokeOperation.aspx.cs"
Inherits="WebUI.Test.InvokeOperation" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title></title>
    <script type="text/javascript">
        function pageLoad()
        {
        }
        function Success(result)
        {
            LabelIsSuccess.innerText += result.IsSuccess;
            LabelError.innerText += result.Error;
            <!-- Changes-->
            if (result.Output_Output1 != null)
                LabelOutput1.innerText += result.Output_Output1;
            if (result.Output_Output2 != null)
                LabelOutput2.innerText += result.Output_Output2;
        }
        function Error(error)
        {
            alert(error._message);
        }
        function Test() {
            var webRequest = Sys.Net.WebServiceProxy.invoke
("http://<servername>/Apriso/WebServices/AJAX/ajaxtest.svc",
          "Invoke", false,
          { "inputs":
                {
          <!-- NAME   : VALUE    -->
                    "Input1": "123",
                    "Input2": "456"
               }
          },
          Success, Error);
        }
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <asp:Button ID="Button1" runat="server" Text="Button" OnClientClick="Test()"/>
            <br /><label ID="LabelIsSuccess" >IsSuccess: </label>
            <br /><label ID="LabelError" >Error: </label>
            <!-- Changes-->
            <br /><label ID="LabelOutput1" >Output1: </label>
```

```
                <br /><label ID="LabelOutput2" >Output2: </label>
            </ContentTemplate>
            </asp:UpdatePanel>
        </div>
        </form>
    </body>
    </html>
```

2.8 PB Web Services Manager

The Web Services Manager contains a list of the Web Services created in the system and provides features for managing them. The manager automatically creates the appropriate database entries and configuration files for the Web Services being created.

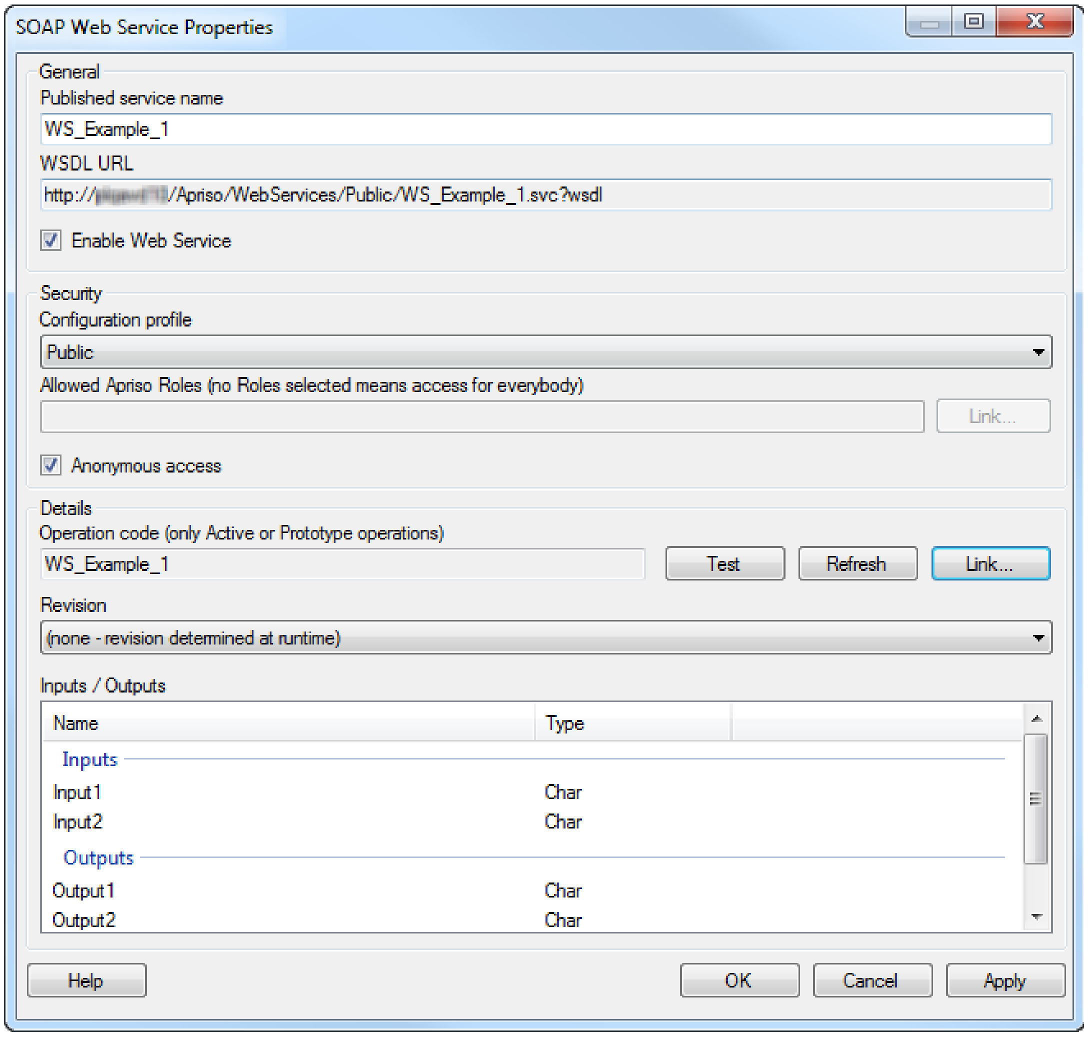


Name	Access Mode	ProjectCode	Status	URL	Definiton URL
Op1_Project_1	Anonymous	Project_1	Enabled	http://[redacted]/Apriso/WebServices/Public/Op1_Project_1.svc	http://[redacted]/Apriso/WebServices/Public/Op1_Project_1.svc?wsdl
APR_ALERT_LIST	Token Authorized		Enabled	http://[redacted]/Apriso/httpServices/operations/APR_ALERT_LIST	http://[redacted]/Apriso/httpServices/operations/APR_ALERT_LIST
APR_CNT_AFTER_HOLD	Anonymous		Enabled	http://[redacted]/Apriso/WebServices/Public/APR_CNT_AFTER_HOLD.svc	http://[redacted]/Apriso/WebServices/Public/APR_CNT_AFTER_HOLD.sv...
APR_CNT_FUTURE_HOLD_PROC...	Token Authorized		Enabled	http://[redacted]/Apriso/httpServices/operations/APR_CNT_FUTURE_HOLD_PRO...	http://[redacted]/Apriso/httpServices/operations/APR_CNT_FUTURE_...
WS_Example_1	Anonymous		Enabled	http://[redacted]/Apriso/WebServices/Public/WS_Example_1.svc	http://[redacted]/Apriso/WebServices/Public/WS_Example_1.svc?wsdl

Figure 1 Web Services Manager

Field	Description
Name	The name of the published Web Service.
Access Mode	There are two types of access modes: Anonymous and Authenticated.
Project Code	The code of the Project that the linked Operation belongs to.
Status	Indicates if the Web Service is enabled or disabled.
URL	The address of the Web Service, which is invoked in runtime.
Definition URL	The URL of the WSDL definition of the Web Service.

Table 1 Web Services Manager columns



SOAP Web Service Properties

General

Published service name: WS_Example_1

WSDL URL: http://[redacted]/Apriso/WebServices/Public/WS_Example_1.svc?wsdl

☑ Enable Web Service

Security

Configuration profile: Public

Allowed Apriso Roles (no Roles selected means access for everybody)

☑ Anonymous access

Details

Operation code (only Active or Prototype operations): WS_Example_1

Revision: (none - revision determined at runtime)

Inputs / Outputs

Name	Type
Inputs	
Input1	Char
Input2	Char
Outputs	
Output1	Char
Output2	Char

Figure 2 Web Service Properties

Field	Description
General	
Published service name	The unique name of the Web Service. This field is required. SOAP published service name must begin with a letter or an underscore.
WSDL URL	The URL of the WSDL definition of the Web Service.
Enable Web Service	Allows the user to temporary disable the Web Service.
Security	
Configuration profile	The name of the security profile (Web server folder) in which the Web Service will be published.
Allowed DELMIA Apriso Roles	The list of DELMIA Apriso Roles which allow invoking the Web Service. An empty list means access for every DELMIA Apriso user. This option is not available for anonymous access.

Anonymous access	If selected, anyone can access the Web Service. Otherwise, two additional Inputs (Username and Password) are published within the Web Service. DELMIA Apriso authentication must be used – a login and password must be provided to invoke the Web Service.
Details	
Operation code	The code of a published Standard Operation. The Operation must be in the Active or Prototype status. This field is required. If the Operation belongs to the Project the Project Code is displayed next to the Operation name (`OperationName (ProjectCode)`).
Revision	The list of Standard Operation revisions from which to choose.
Test button	Opens the Test Web Service window. The user has the ability to invoke a published Web Service with test parameters.
Refresh button	Reloads the external Inputs and Outputs of the Standard Operation.
Link button	Displays a list of Standard Operations to use.
Input/Output list	The list of external Inputs and Outputs of the published Standard Operation.

Table 2 Web Service Properties

2.9 Quick Start

It is possible to publish an Operation as a Web Service using WS Manager in Process Builder. After publishing, the Web Service can be called by any external application.

The following steps should be performed to publish an Operation as a Web Service:

1. Open Web Services Manager from the View menu.
2. Select New SOAP Web Service from the context menu. The system displays a window that allows for specifying the Web Service properties.
3. Click the Link button and choose a Standard Operation from the list (or optionally choose a revision from the drop-down list).

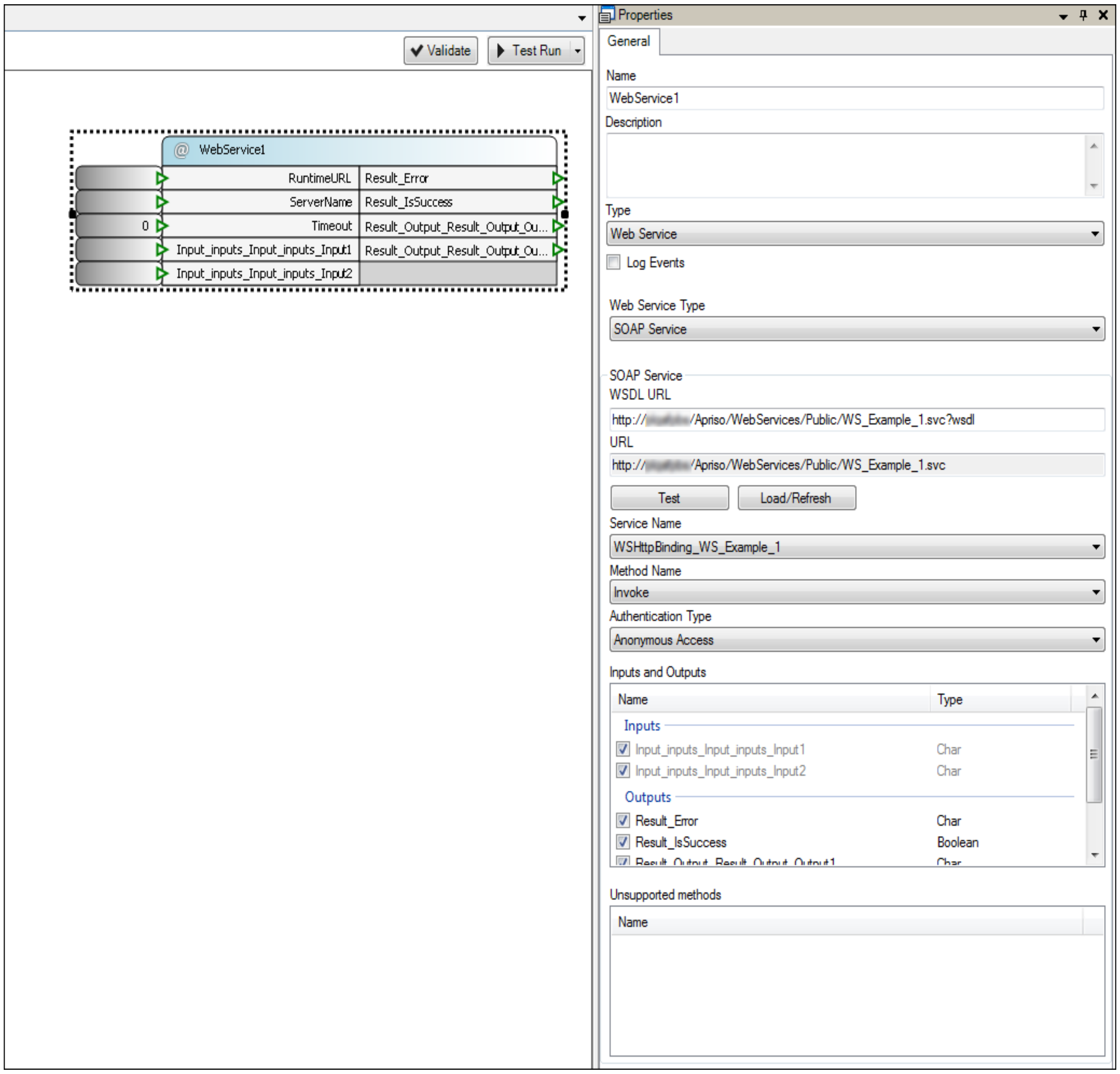
Apply the settings. The system will publish the Operation as a Web Service.

3 Configuration of Invoking Web Services in Process Builder

3.1 Web Service PB Function

Process Builder allows for executing remote Web Services from PB Operations using a Web Service Function. In this Function, the user may configure which Web Service is to be invoked and what parameters are to be passed and retrieved from it.

In this version of Process Builder, only simple types of Web Service parameters are supported. Methods using data types (e.g., methods with Output parameters in method signatures: `void MyMethod(out string text)`) are listed as "unsupported" in the Web Service Function properties.



WebService1

Inputs	Outputs
RuntimeURL	Result_Error
ServerName	Result_IsSuccess
0 Timeout	Result_Output_Result_Output_Ou...
Input_inputs_Input_inputs_Input1	Result_Output_Result_Output_Ou...
Input_inputs_Input_inputs_Input2	

Name: WebService1

Description:

Type: Web Service

Log Events

Web Service Type: SOAP Service

SOAP Service

WSDL URL: http://[redacted]/Apriso/WebServices/Public/WS_Example_1.svc?wsdl

URL: http://[redacted]/Apriso/WebServices/Public/WS_Example_1.svc

Service Name: WSHttpBinding_WS_Example_1

Method Name: Invoke

Authentication Type: Anonymous Access

Inputs and Outputs

Name	Type
Inputs	
☑ Input_inputs_Input_inputs_Input1	Char
☑ Input_inputs_Input_inputs_Input2	Char
Outputs	
☑ Result_Error	Char
☑ Result_IsSuccess	Boolean
☑ Result_Output_Result_Output_Output1	Char

Unsupported methods

Name

Figure 3 Web Service Function Properties

In the Properties pane, there are the parameters of the Web Service Function, which the user is able to manage.

Field	Description
WSDL URL	The WSDL definition of the Web Service.
URL	The URL of the WSDL definition of the Web Service.
Test button	Performs a test execution of the Web Service defined by the URL.
Load/Refresh button	Reads the WSDL definition of the Web Service containing the list of services, methods, and their Inputs and Outputs.
Service Name	The name of the remote service which will be invoked.
Method Name	The name of the method which will be invoked.
Authentication Type	There are three types of Authentication from which to choose:

	▶ Anonymous Access – no authentication ▶ Basic Authentication – HTTP basic authentication (the Username and Password Inputs are added to the Function and used to access the Web Service) ▶ NTLM Authentication – the authentication protocol used in the Microsoft Windows networks
Inputs and Outputs	The list of Inputs and Outputs used to invoke the remote Web Service.
Unsupported methods	The list of methods with parameters of complex types (that are not supported by Process Builder).

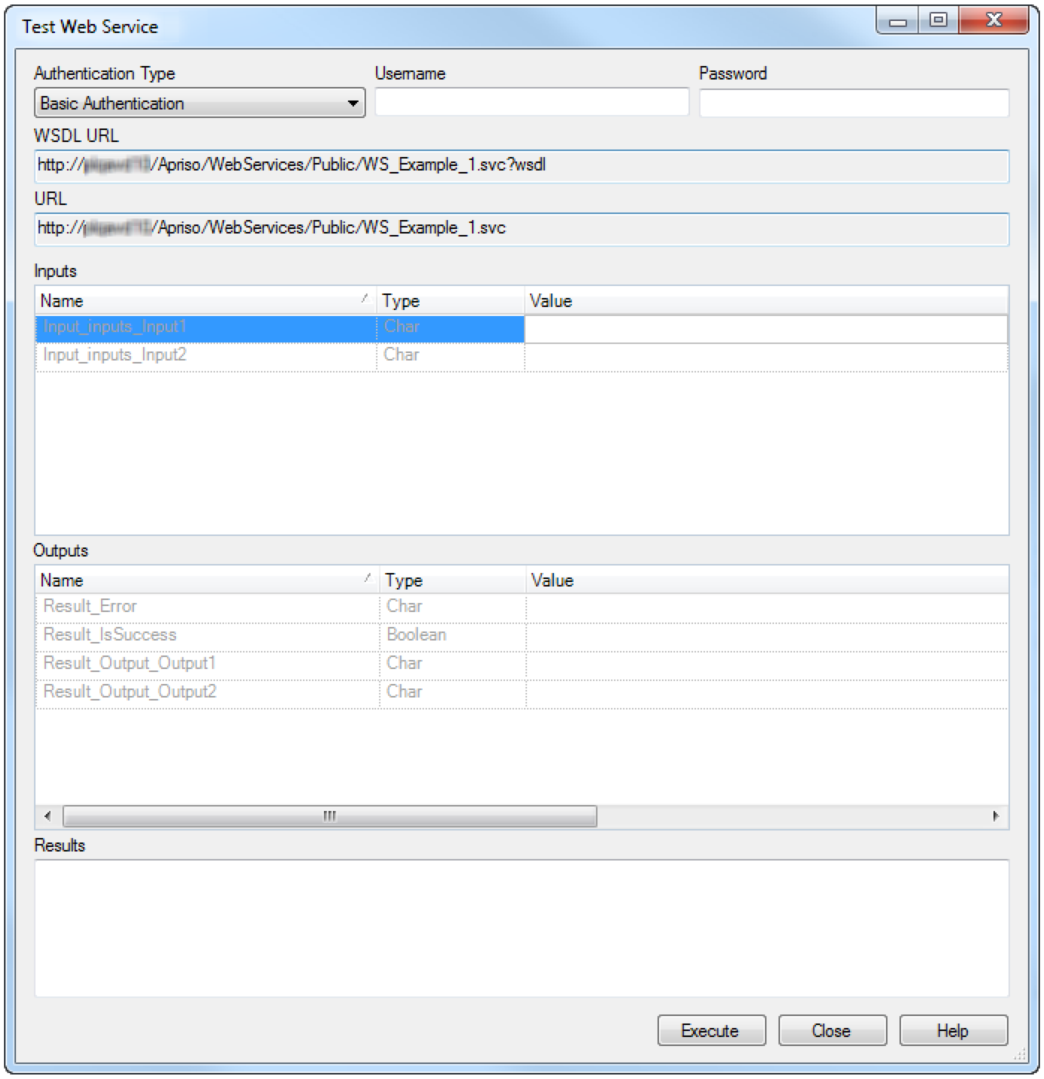
Table 3 Web Service Function Properties

After choosing the Web Service method to invoke, the system determines the Web Service prototype based on its WSDL and the method name, and it then creates the needed Inputs and Outputs of the Function.

Inputs	Description
RuntimeURL	The URL of a Web Service which is used to call the Web Service. This overrides the URL from the WSDL definition when specified.
ServerName	The server name which is used to call a Web Service. If specified, this overrides the host name part of the URL from the WSDL definition.
Timeout	The connection timeout for calling a Web Service in runtime. A zero value means using the default timeout value (one minute).
Username	The name of the user that is used for authentication.
Password	The password used for authentication.

Table 4 Web Service Function Inputs

It is possible to test a remote Web Service by pressing the Test button on the Web Service Function Properties pane. A new window named Test Web Service will appear, where the user may enter the values for the Inputs and then execute the remote WS method.



Authentication Type: Basic Authentication
Username:
Password:
WSDL URL: http://[redacted]/Apriso/WebServices/Public/WS_Example_1.svc?wsdl
URL: http://[redacted]/Apriso/WebServices/Public/WS_Example_1.svc

Inputs

Name	Type	Value
Input_inputs_Input1	Char	
Input_inputs_Input2	Char	

Outputs

Name	Type	Value
Result_Error	Char	
Result_IsSuccess	Boolean	
Result_Output_Output1	Char	
Result_Output_Output2	Char	

Results

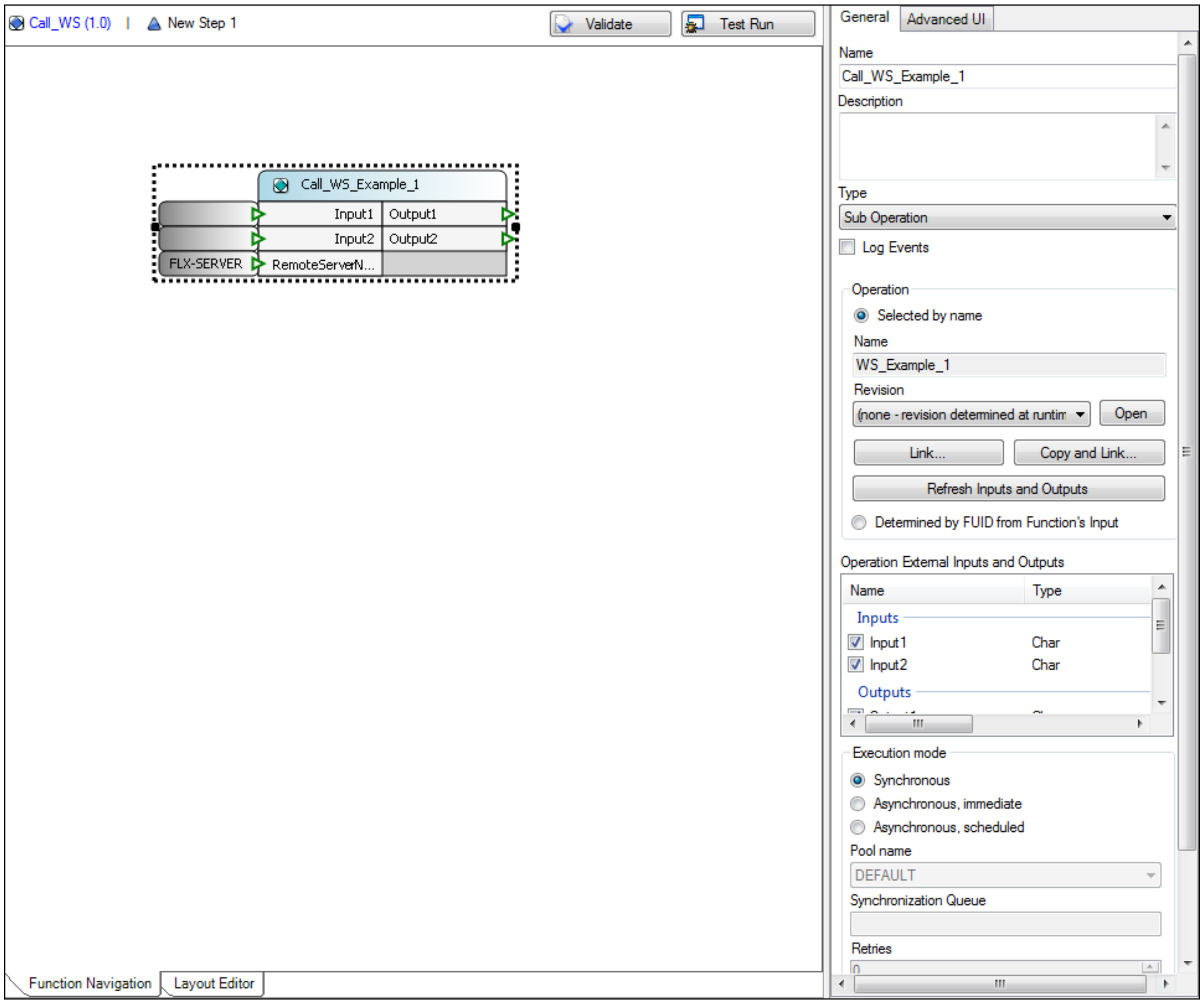
Figure 4 Test of Web Service

Field	Description
Authentication Type	There are three types of Authentication from which to choose: ▶ Anonymous Access – no authentication ▶ Basic Authentication – HTTP basic authentication (the Username and Password text boxes are shown on the screen) ▶ NTLM Authentication – the authentication protocol used in Microsoft Windows networks
WSDL URL	The URL of the WSDL definition of the Web Service.
URL	The address of the Web Service which will be invoked in runtime.
Inputs field	The Inputs of a remote WS method.
Outputs field	The Outputs of a remote WS method.
Results	The results of invoking a Web Service with test parameters.
Execute button	Executes a test of the remote Web Service method.

Table 5 Test Web Service window

3.2 DELMIA Apriso to DELMIA Apriso Communication

It is possible to invoke Operations from a remote DELMIA Apriso server using an embedded feature of a Sub Operation Function.



Name: Call_WS_Example_1

Description:

Type: Sub Operation

Log Events

Operation

(●) Selected by name

Name: WS_Example_1

Revision: (none - revision determined at runtim

( ) Determined by FUID from Function's Input

Operation External Inputs and Outputs

Name	Type
Inputs	
☑ Input1	Char
☑ Input2	Char
Outputs	

Execution mode

(●) Synchronous

( ) Asynchronous, immediate

( ) Asynchronous, scheduled

Pool name: DEFAULT

Synchronization Queue:

Retries: 0

Figure 5 Remote Sub Operation invocation

The user can choose another remote DELMIA Apriso server by selecting the **Remote** check box on the Sub Operation Function properties pane. A new Input (type Char) called RemoteServerName is created, and the name of the remote server should be passed as the value of this Input.

In the synchronous mode, the Sub Operation Function invokes a dedicated Web Service on a remote machine under the following URL:

```
http://<server name>/Apriso/WebServices/FlexNetOperationsService.svc
```

In the asynchronous mode, the local Job Scheduler receives a task to call the above Web Service in the background.

On the remote server, the IIS directory security settings and the `web.config` file for the FlexNetOperationsService must be configured to allow incoming connections:

```
<service behaviorConfiguration="FlexNet.WebUI.WebServices.FlexNetOperationsServiceBehavior"
name="FlexNet.WebUI.WebServices.FlexNetOperationsService">
    <endpoint address="" binding="wsHttpBinding"
bindingConfiguration="FlexNetConfiguration"contract="FlexNet.WebUI.WebServices.IFlexNetOper
ationsService">
        <identity>
            <dns value="localhost" />
        </identity>
    </endpoint>
    <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
</service>
```

By default, the section above (in the `web.config` file) is put in comments. To enable remote Operation execution on a destination server, the section should be uncommented.

4 References

Internal Documentation

1. ***Process Builder Help***

   Provides an overview of DELMIA Apriso Process Builder (PB) and information on installing and using the application. This Help describes the user interface elements, entity maintenance, available Business Controls, and management of Processes, Operations, and Screen Flows.

2. ***Security Implementation Guide***

   Provides an overview of DELMIA Apriso security and information on effectively securing all instances of DELMIA Apriso.

3. ***Central Configuration Documentation***

   Describes in detail all the keys of the Central Configuration (CC) file for DELMIA Apriso. Various sections group the keys for individual modules or distinct functional areas.

External Documentation

1. *Windows Communication Foundation Architecture Overview* at Microsoft Docs
2. *Windows Communication Foundation* at Microsoft Docs

3DS Support Knowledge Base

If you have any additional questions or doubts not addressed in our documentation, feel free to visit the **3DS Support Knowledge Base** at https://support.3ds.com/knowledge-base/.