



Robotic Process Automation Software

Web Services

USER GUIDE

Major Version: 6

Document Revision: 1.0

For more information please contact:

info@blueprism.com | UK: +44 (0) 870 879 3000 | US: +1 888 757 7476

www.blueprism.com

Contents

1. Introduction	3
1.1. Audience.....	3
2. Executive Summary	4
3. Exposing Web Services	5
3.1. Providing Access to the Web Services	6
3.2. Defining Web Service Parameters	7
3.3. Web Service Encoding Types	7
3.4. Walkthrough: Expose a Business Object as a Web Service.....	8
3.5. Walkthrough: Expose a Process as a Web Service.....	11
4. Consuming Third-Party Web Services	14
4.1. Accessing Third-Party Web Services	15
4.2. Web Service Parameters	16
4.3. Walkthrough: Setup a Third-Party Web Service in Blue Prism.....	18
4.4. Walkthrough: Use a Third-Party Web Service in Blue Prism.....	22
5. Advanced Topics.....	26
5.1. Design Considerations for a Blue Prism Web Service Interface	26
5.2. Securing Exposed Blue Prism Web Services.....	34
5.3. Consume Web Services using Code Stage Based Business Objects	35
5.4. Consume an Exposed Blue Prism Web Service using Visual Studio	45
6. Frequently Asked Questions.....	54
7. Support.....	55

The information contained in this document is the proprietary and confidential information of Blue Prism Limited and should not be disclosed to a third party without the written consent of an authorised Blue Prism representative. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying without the written permission of Blue Prism Limited.

© Blue Prism Limited, 2001 – 2016

®Blue Prism is a registered trademark of Blue Prism Limited

All trademarks are hereby acknowledged and are used to the benefit of their respective owners.

Blue Prism is not responsible for the content of external websites referenced by this document.

Blue Prism Limited, Centrix House, Crow Lane East, Newton-le-Willows, WA12 9UY, United Kingdom

Registered in England: Reg. No. 4260035. Tel: +44 870 879 3000. Web: www.blueprism.com

1. Introduction

This document provides an overview of the capabilities of Blue Prism when used in environments which feature Web Services.

This guide provides information about:

- Exposing a Blue Prism Business Object as a web service.
- Exposing a Blue Prism Process as a web service.
- Consuming third-party web services in a Blue Prism Business Object or Process.

The following advanced topics are also included:

- Design considerations for a Blue Prism Web Service interface
- Expose Blue Prism Web Services using a Proxy
- How to interact with third-party web services that provide data using a polymorphic structure.
- How to create an independent web service client in Visual Studio that can interact with an exposed Blue Prism web service.

1.1. Audience

This document is intended to provide information to solution architects and designers, developers and Blue Prism accredited developers.

2. Executive Summary

Blue Prism is designed to automate any application that can be accessed from a Windows PC through the Graphical User Interface (GUI), however there are some scenarios where users may wish to automate and integrate via web services.

Web service connectivity is supported by Blue Prism to allow:

- Third-party applications or developers to utilise and initiate Business Objects and Process within Blue Prism for purposes of:
- Interacting with legacy systems which are already automated by Blue Prism.
- Adding work items to the appropriate queues ready for processing based on the pre-determined schedules.
- Triggering work items to be processed by Blue Prism immediately (subject to resource availability).
- Retrieving data from systems which Blue Prism automates.
- Retrieving information from Blue Prism (e.g. details of work queues, schedules, work history etc.).
- Blue Prism Business Objects and Processes to interact with third-party systems through use of published web services.

Web service integration is a traditional software development technique and its use should be evaluated against the following high-level considerations:

- Performance impact on underlying systems during periods of high demand.
- Ensuring that any data validation that is implemented at the presentation layer is manually applied to web service interactions (e.g. verifying that this validation logic is not bypassed through use of web services).
- The level of IT governance may differ to that required for GUI-based automations. Often web services provide broader capabilities than can be achieved by an end user working through the graphical interface of an application.

3. Exposing Web Services

Once implemented, Blue Prism quickly becomes a secure repository that contains detailed information about the various systems and business processes that are available to be processed and worked by the virtual workforce provided by Blue Prism.

This information contains all of the available actions that can be taken across the various systems and technologies, as well as the details of what rules, decisions and procedures need to be followed in order for the Blue Prism runtime resources to successfully complete a process.

Through exposing the Blue Prism Business Objects and Processes, third-party systems and developers can invoke these web services to utilise the power and flexibility of Blue Prism for a number of purposes:

- Interacting with legacy systems which are already automated by Blue Prism.
- Adding work items to the appropriate queues ready for processing based on the pre-determined schedules.
- Triggering work items to be processed by Blue Prism immediately (subject to resource availability).
- Retrieving data from systems which Blue Prism automates.
- Retrieving information from Blue Prism (e.g. details of work queues, schedules, work history etc.).

The approach of allowing third-party systems or developers to interact with Blue Prism in this way provides a number of advantages:

- All third-party systems and developers connect to a given system via a method which enforces the rules designed into Blue Prism.
- Enforces commonality in the way that systems are automated (e.g. removes different rules and paths being followed by each developer or system).
- Blue Prism provides a common interface for any system that it automates irrespective of the technology that is being automated (i.e. mainframe, web applications, .NET application, excel, SQL database etc.)
- Provides a common method of applying any changes required to the interface, and aligns them with existing internal change management policies. Additionally all changes to the actions and processes required are managed and controlled centrally using the visual designers by the appropriate team(s).

Blue Prism web services are natively SOAP, WSDL-based services. Native support is not provided for creating RESTful Blue Prism web services.

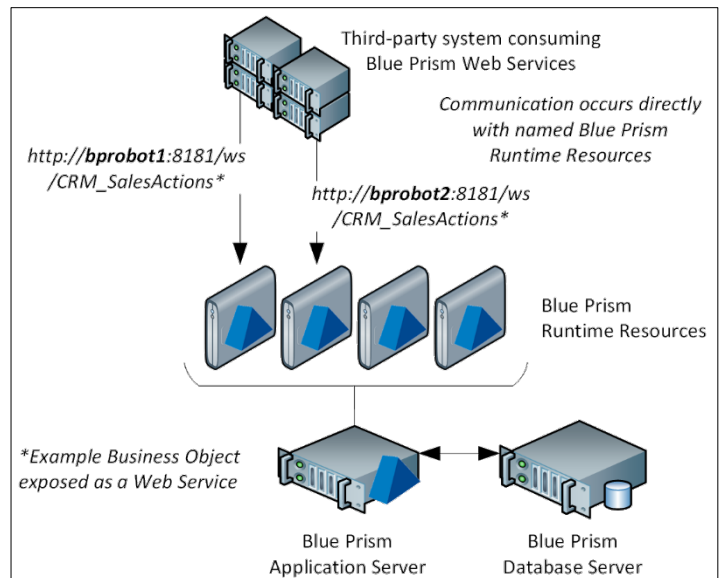
3.1. Providing Access to the Web Services

In order to allow access to Blue Prism web services the following items will need to be reviewed:

- Network connectivity to the Blue Prism runtime resources**
 The systems which are to interact with the web services will need to have network connectivity to the designated Blue Prism resources.

By default Blue Prism web services are made available on port 8181 for TCP traffic however this is configurable.

See **Securing Exposed Blue Prism Web Services** for further information about securing these connections using certificate-based encryption.



- Security permissions to interact with the web services**
 When a system makes a call to a Blue Prism web service, Blue Prism access credentials will need to be provided that give the system or user the appropriate permissions to carry out the actions.

It is recommended that a user account is set up for each third-party provider that will interact with the web services. This user account should be allocated the minimum security permissions required to successfully execute the Blue Prism Business Objects and Processes. (E.g. if the process requires access to credential information that is restricted, the account used by the third-party system will require appropriate permissions to utilize this credential).

Where Blue Prism is integrated with Active Directory for Single Sign-on, the connecting system will need to provide the credentials for a domain account with appropriate Blue Prism privileges.

- Web service definition (WSDL)**
 The third-party system or developer will typically require access to the WSDL for the appropriate web service(s). Once the machine name and port for the runtime resource have been identified the following URL will provide the name of each available web service, a short description and the address of the WSDL. The WSDL contains the path of the web service.

`http://[machine name]:[port]/ws/`

Where the Blue Prism Runtime Resource has been configured with certificate-based encryption, the prefix will be **https**.

3.2. Defining Web Service Parameters

The Blue Prism Web services can be configured to both accept input parameters, and return output parameters. The parameters that will be configured as part of the Blue Prism web services are based on the input and output parameters defined on the respective Business Objects and Processes.

The displayed Blue Prism data types are supported for both input and output parameters and will be mapped to the appropriate web service XSD data types.

Blue Prism supports the use of data collections (lists) as both input and output parameters providing that the structure of the collection is predetermined and is made up exclusively of data items of a supported data type.

Blue Prism Data Type	Web Service XSD Data Type
date	date
dateTime	dateTime
flag	boolean
number	decimal
text	String
time	time
timespan	duration
binary	base64binary

3.3. Web Service Encoding Types

The encoding type that will be used to present a given web service is based on the nature of the input and output parameters that are defined.

By default a Blue Prism web service will be exposed as a **RPC/Encoded** web service, however if any input or output parameters of the web service are configured to leverage a Blue Prism Data Collection, then those actions will be exposed as **Document/Literal** encoded operations.

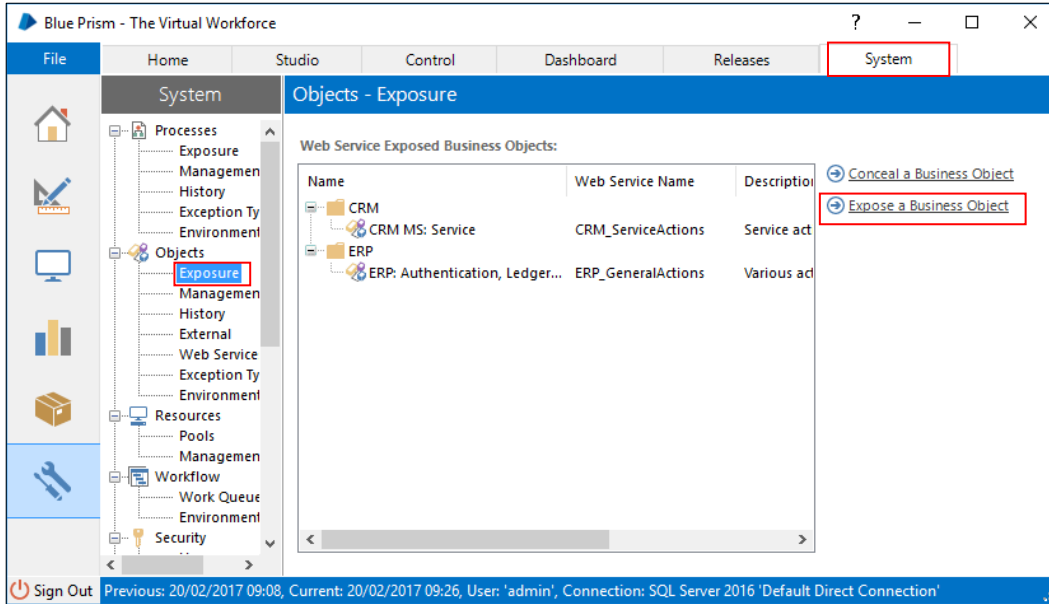
Blue Prism Business Objects automatically include a number of default actions which will be encoded as **RPC/Encoded** operations meaning that the WSDL for an exposed Business Object will always include at least one **RPC/Encoded** operation.

Exposed Blue Prism web services are presented with **UTF-8** character encoding.

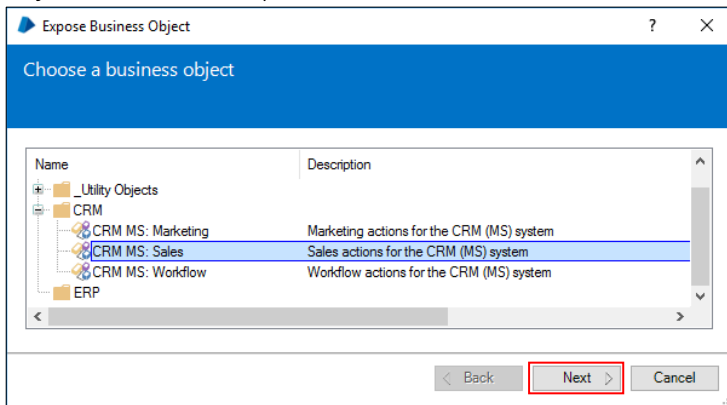
3.4. Walkthrough: Expose a Business Object as a Web Service

This walkthrough provides the steps required to make a Blue Prism Business Object available as a web service. Steps are also provided to verify that a Business Object is available as a web service and how to access the WSDL.

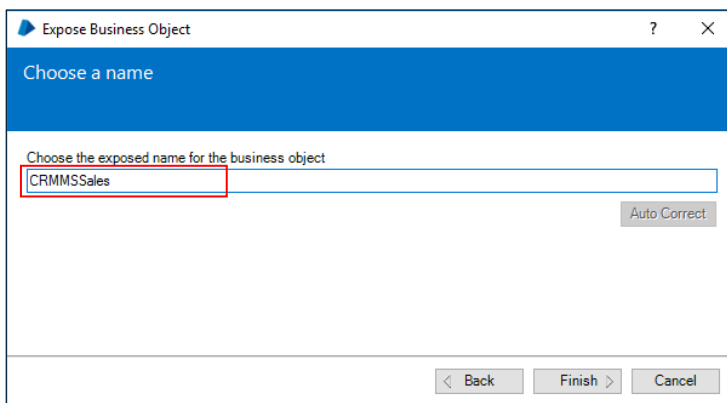
1. Enter **System Manager** and select the **Objects** area and click **Expose a Business Object**



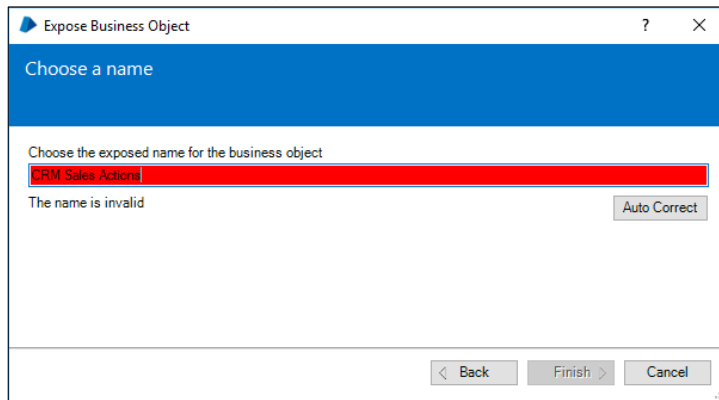
2. A list of Business Objects which are not currently exposed as web services is presented. Select the Business Object that is to be exposed and click **Next**.



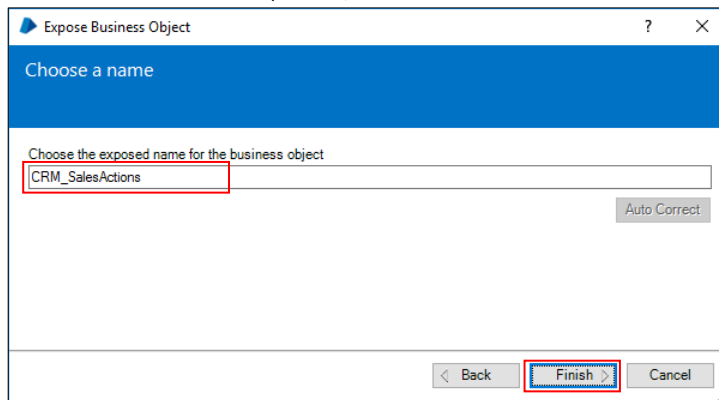
3. The wizard will use the name of the Business Object to suggest a suitable name that will be used for the web service.



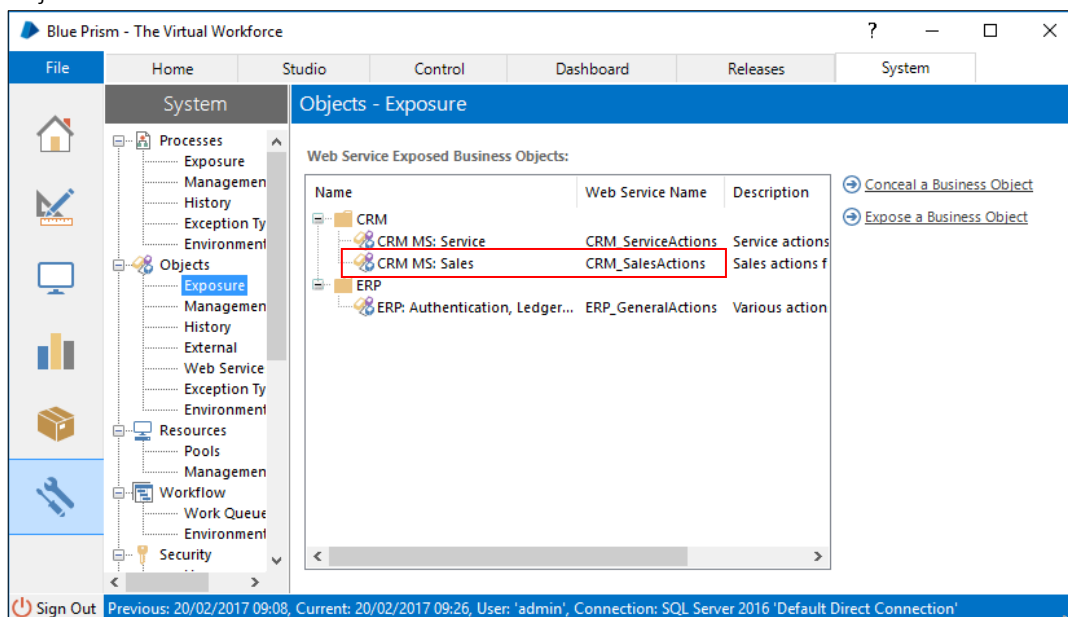
It is possible to change the name at this point however if the name is a duplicate of a previously exposed Blue Prism Web Service or if any invalid characters are detected a warning will be shown. Clicking Auto Correct will remove any invalid characters and ensure that the name is unique.



4. Once the name is acceptable, Click Finish.

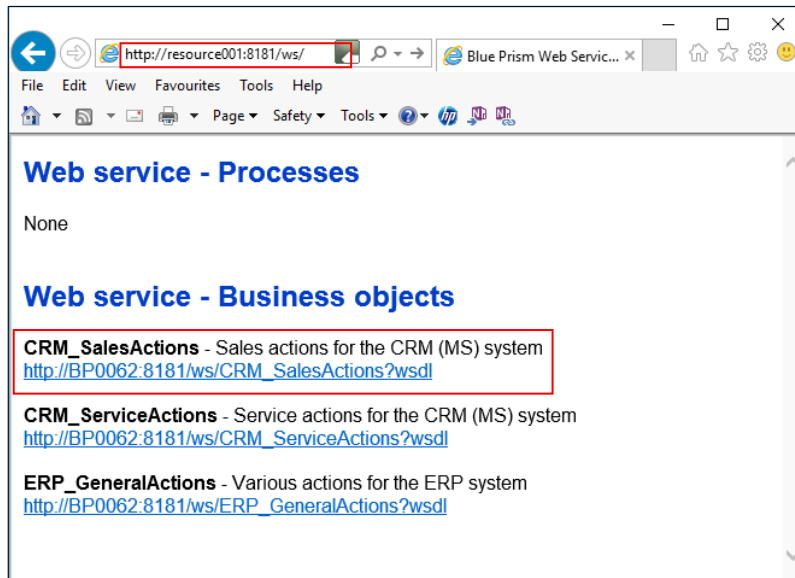


5. The newly exposed web service name will now appear in the list alongside the name of the Business Object.



- The web services that are exposed from Blue Prism can be confirmed by using the following URL which also provides the address of the WSDL.
`http://[machinename]:<port>/ws/`.

In this example the machine name is **resource001** and the default port of **8181** is in use:
`http://resource001:8181/ws/`

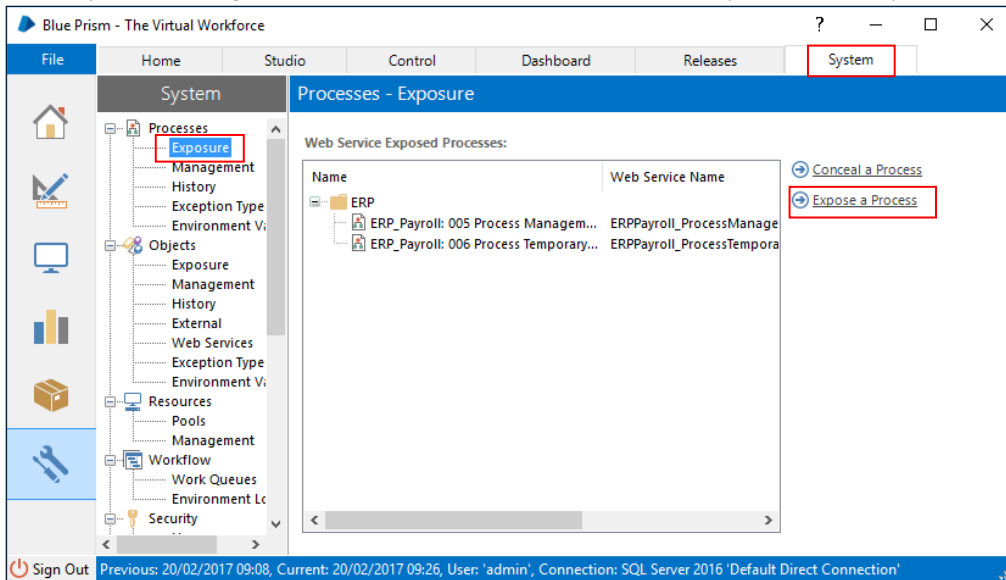


Where the Blue Prism Runtime Resource has been configured with certificate-based encryption, the prefix will be **https**.

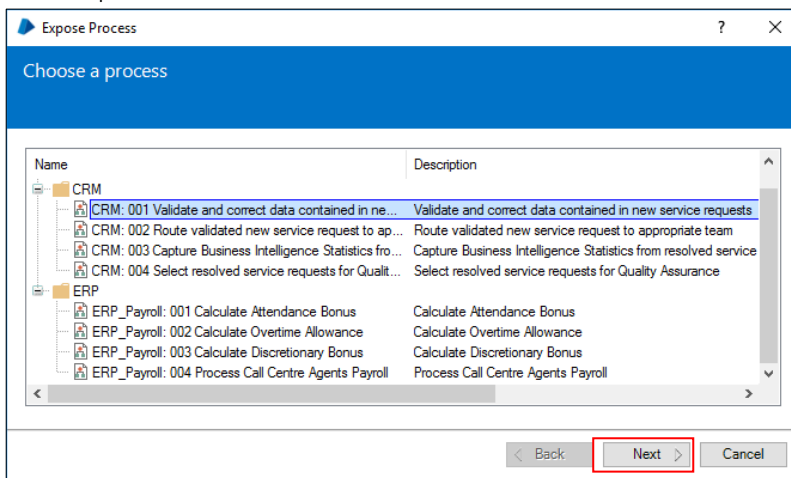
3.5. Walkthrough: Expose a Process as a Web Service

This walkthrough provide the steps required to make a Blue Prism Process available as a web service. Steps are also provided to verify that a Process is available as a web service how to access the WSDL.

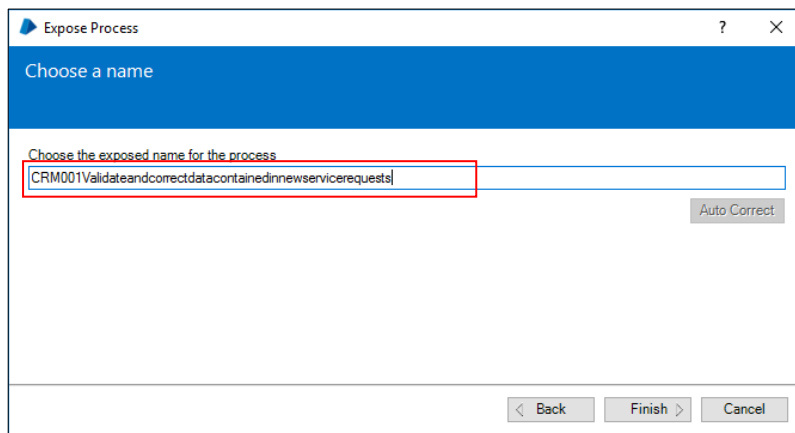
1. Enter **System Manager** and select the **Processes** area. Click **Exposure** and **Expose a Process**.



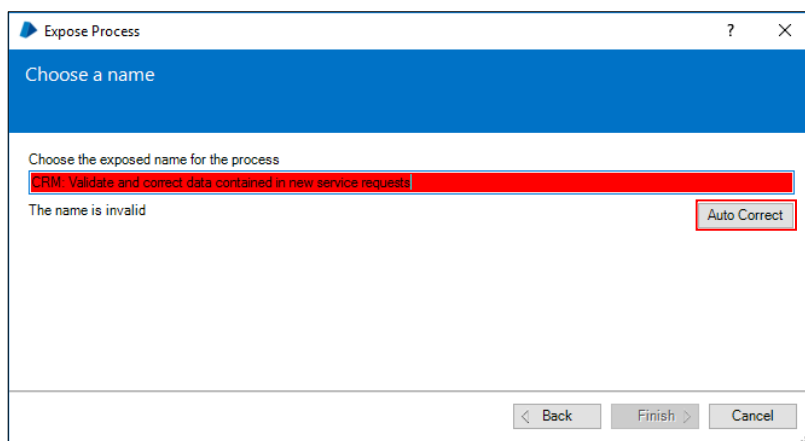
2. A list of Processes which are not currently exposed as web services is presented. Select the Process that is to be exposed and click **Next**.



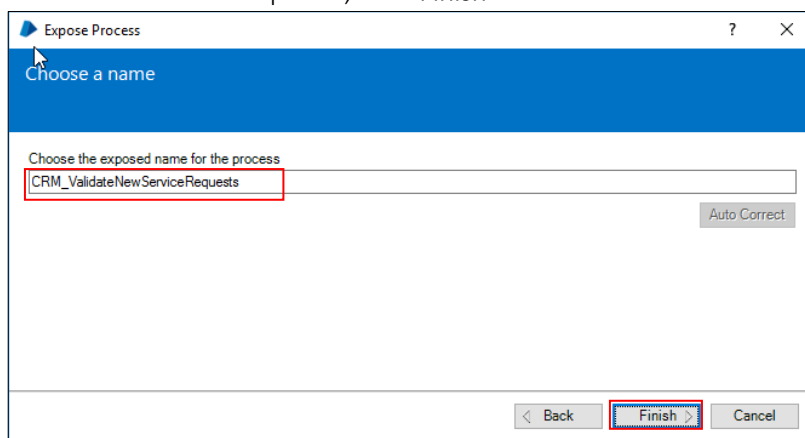
- The wizard will use the name of the Process to suggest a suitable name that will be used for the web service.



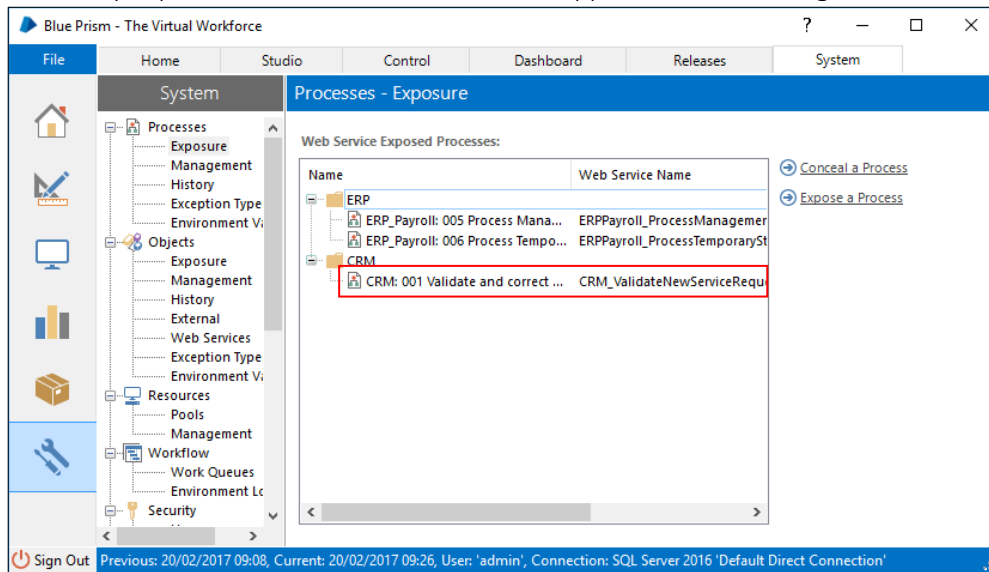
- It is possible to change the name at this point however if the name is a duplicate of a currently exposed Blue Prism web service or if any invalid characters are detected, a warning will be shown. Clicking Auto Correct will remove any invalid characters and ensure that the name is unique.



- Once the name is acceptable, Click **Finish**.

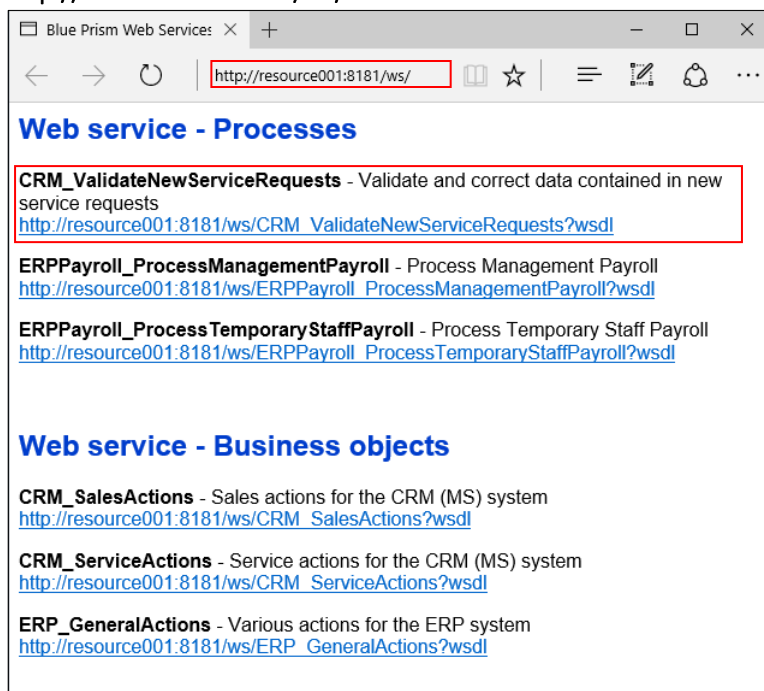


- The newly exposed web service name will now appear in the list alongside the name of the Process.



- The web services that are exposed from Blue Prism can be confirmed by using the following URL which also provides the address of the WSDL.
[http://\[machinename\]:<port>/ws/](http://[machinename]:<port>/ws/).

In this example the machine name is **resource001** and the default port of **8181** is in use:
<http://resource001:8181/ws/>



Where the Blue Prism Runtime Resource has been configured with certificate-based encryption, the prefix will be **https**.

4. Consuming Third-Party Web Services

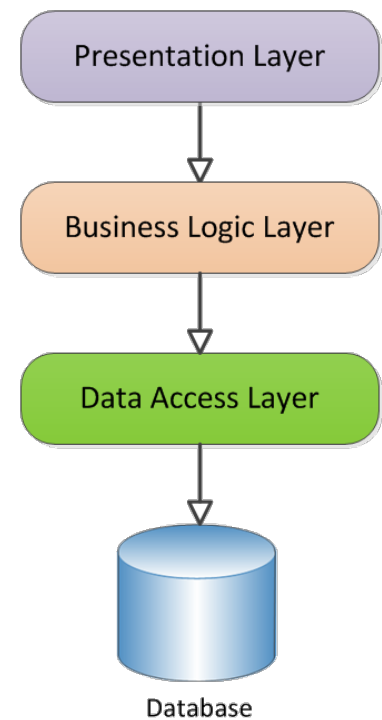
Blue Prism provides the ability to automate third-party applications through the use of published and accessible web services.

An intuitive wizard assists users to configure Blue Prism to be aware of any number of web services and once setup, the configured web services are available for use within the Blue Prism visual designer tools (Object Studio and Process Studio). These web services are presented and utilised in a way that is code free and intuitive.

Unlike typical Blue Prism automations which interact with applications using the Graphical User Interface (GUI), web service integration is a traditional technique that operates at the Business Logic layer (as opposed to the presentation layer) and maybe susceptible to additional complexity.

The following topics should be reviewed when considering the use of web service integration:

- It is common for applications to apply more stringent data quality and validation rules at the **Presentation Layer** than the **Business Logic Layer**. Web services interact directly with the **Business Logic Layer** and therefore it is possible that validation logic that is only presented via the GUI will be bypassed. Therefore it may be necessary to manually implement additional verification steps to ensure such validation continues to be applied.
- Web services may provide processes with a wider number of actions in the third-party systems than the users can achieve via the user interface and therefore may require additional IT governance.
- By directly connecting with the **Business Logic Layer** additional strain may be placed on the platforms that underpin the target applications. It may be necessary to add additional configuration to the processes to ensure that the target applications are not over-burdened during periods of high-use or demand.
- There are a large number of technologies, conventions and standards available for publishing web services - therefore is not possible to natively support all of these within Blue Prism. Where a web service does not conform to the standards supported within Blue Prism it will be necessary to use programming techniques to interface with web service(s).



4.1. Accessing Third-Party Web Services

Blue Prism can consume third-party web services which are SOAP-based with the following binding styles:

- RPC/encoded
- document/literal

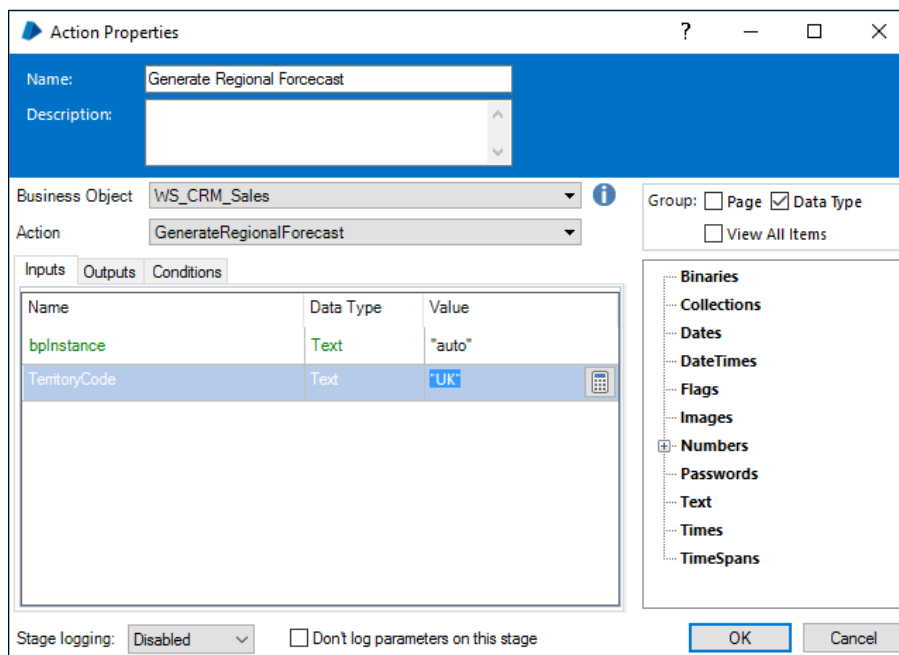
The ability to consume such web services is also reliant on:

- The input/output web service data types having a corresponding data type within Blue Prism (as described above)
- The web service WSDL being valid and wholly containing compliant XML
- Blue Prism having appropriate connectivity to the web service
- The character encoding of the web service being UTF-8
- Access authentication being handled via one the of the following methods:
 - Independently of the web service call (e.g. based on IP security).
 - Through use of mutual, certificate based authentication.
 - Through use of username and password as part of the HTTP header.

Support is primarily provided for a range of SOAP Web Services, however support for REST web services is provided for developers with appropriate programming experience through use of a series of example VBOs: REST, HTTP, JSON and Oauth.

In order to access a third-party web service from within Blue Prism, the SOAP WSDL of the respective web service can be imported via System Manager. (See **Walkthrough: Setup a Third-Party Web Service in Blue Prism** for guidance).

Once setup, a web service is available via Object Studio or Process Studio as if accessing a pre-existing Blue Prism Object.



4.2. Web Service Parameters

Blue Prism can pass data as parameters to the web services which will define the parameters that are available to be passed along with the type of data that is expected. The following web service (XSD) data types are supported by Blue Prism.

Web Service XSD Data Type	Blue Prism Data Type
date	date
dateTime	datetime
boolean	flag
decimal, float, double, pDecimal, integer, nonPositiveInteger, nonNegativeInteger, long, negativeInteger, unsignedLong, int, positiveInteger, unsignedInt, short, unsignedShort, byte, unsignedByte	number
String, normalizedString	text
time	time
duration	timespan
base64binary	binary

If a web service to be consumed has any input or output parameters which have been defined with a type other than one presented in the above table, this web service is not natively compatible with Blue Prism.

4.2.1. Complex Data Types

When consuming web services, complex types (e.g. compound structures and arrays, lists) that are composed of the supported types listed above can also be consumed. They will be mapped into Collections within Blue Prism.

Native support is provided for a subset of the complex data type schema constructions including: **xsd:all**, **xsd:element**, and **xsd:sequence**.

When exposing complex types (collections) via input or output parameters, the collections must be set up within Blue Prism with pre-defined fields (rather than a dynamically defined structure).

Web services which use polymorphic data structures as input or output parameters are not natively compatible with Blue Prism.

Where data structures are not natively supported, advanced techniques involving code stages can typically be used to work with such services. (See **Consume Web Services using Code Stage Based Business Objects**).

4.2.2. WS-Security: Passing Credentials in the SOAP Header

Microsoft Web Service Enhancements (WSE) 3.0 includes a set of classes that implement additional web service specifications to the SOAP protocol such as custom security and reliable messaging. These specifications are commonly denoted by the **WS-** prefix and includes **WS-Security**.

WS-Security implemented by WSE 3.0 requires that the web service authentication credentials (username and password) are passed within the SOAP header.

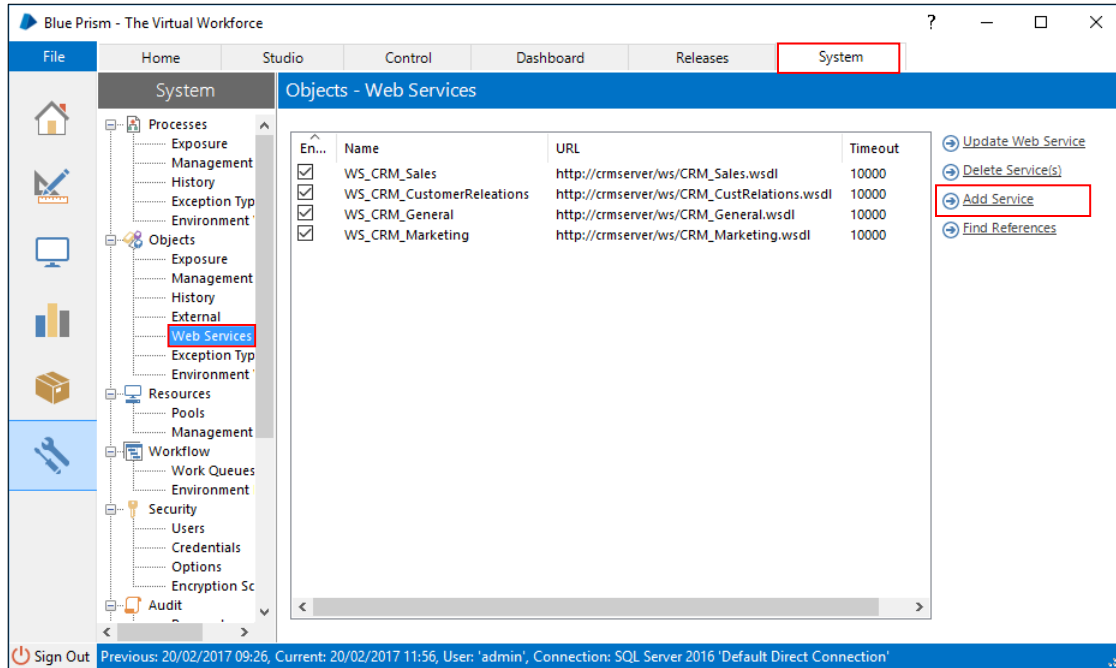
When consuming web services with this configuration it is necessary to use advanced techniques including code stages. (See **Consume Web Services using Code Stage Based Business Objects**).

Related configuration can also be achieved using code stages and may include specifying the value of attributes such as **MustUnderstand** or configuring whether to include the WS-Addressing headers.

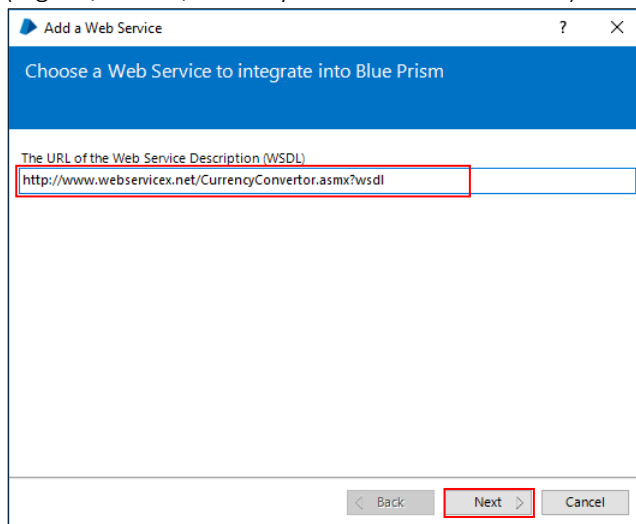
4.3. Walkthrough: Setup a Third-Party Web Service in Blue Prism

This walkthrough provide the steps required to make a third-party web service available within Blue Prism. See the separate walkthrough for instructions on how to use/consume the third-party web service.

1. Enter **System** and select **Web Services** within the **Objects** area. Click **Add Service** to launch the wizard.



2. Enter the **URL** of the WSDL which defines the web service and click **Next**. If the WSDL is stored locally, it is acceptable to enter a local file location (e.g. c:\WSDLs\CurrencyConversionService.wsdl).



- If required by the web service, enter the **authentication information** for the web service. Click **Next**.

The screenshot shows the 'Add a Web Service' dialog box with the following details:

- Title: Add a Web Service
- Header: Please supply authentication details for the web service.
- Option 1: Web Service Requires Username and password (HTTP Authentication)
 - Username: serviceUsername
 - Password: [Redacted]
- Option 2: Web Service Requires client side certificate (SSL Authentication)
 - Store Location: [Dropdown]
 - Store Name: [Dropdown]
 - Find Type: [Dropdown]
 - Find Criteria: [Text Box]
 - Password: [Text Box]
- Buttons: Back, Next, Cancel

- Specify the timeout value – this is the length of time (milliseconds) which Blue Prism will wait for a response from the web service before assuming the request has timed out. Click **Next**.

The screenshot shows the 'Add a Web Service' dialog box with the following details:

- Title: Add a Web Service
- Header: Please set the timeout to use when interacting with the web service
- Text: The timeout to use (in milliseconds) when interacting with the service
- Input: 10000
- Buttons: Back, Next, Cancel

- Blue Prism will parse the information contained within the provided WSDL. Click **Next** to proceed.

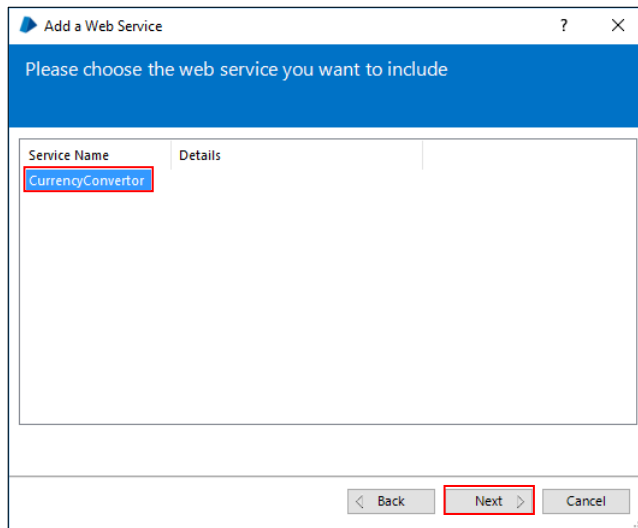
The screenshot shows the 'Add a Web Service' dialog box with the following details:

- Title: Add a Web Service
- Header: Importing web service definitions.
- Text: Retrieved wsdl from http://www.websvc.net/CurrencyConverter.asmx?wsdl
- Log Output:

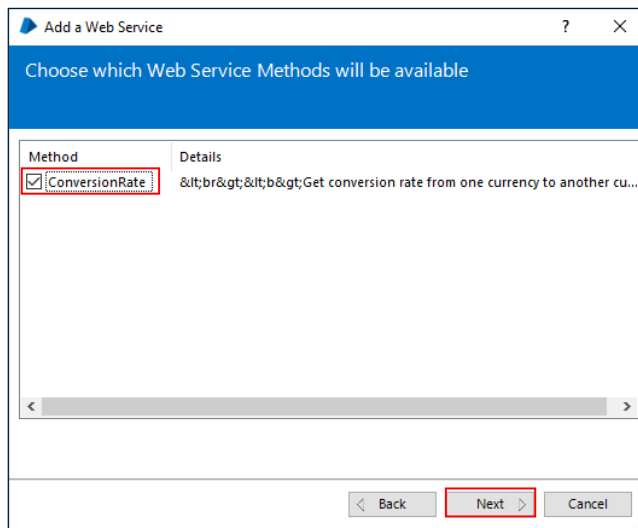

```

Found service 'CurrencyConverter'
Found service port 'CurrencyConverterSoap'
Found operation 'ConversionRate'
Found message 'ConversionRateSoapIn'
Found part 'FromCurrency'
Found part 'ToCurrency'
Found message 'ConversionRateSoapOut'
Found part 'ConversionRateResult'
Found service port 'CurrencyConverterSoap12'
Found service port 'CurrencyConverterHttpGet'
Found service port 'CurrencyConverterHttpPost'
SUCCESS: Retrieved all web service details
            
```
- Buttons: Back, Next, Cancel

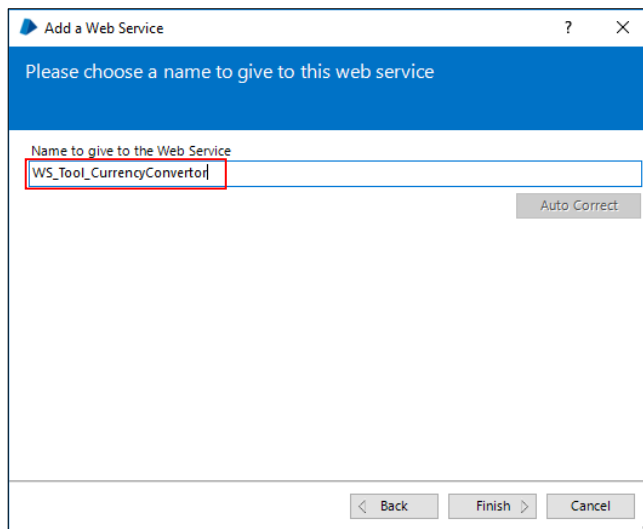
6. Select the web service that is to be consumed and click **Next**.



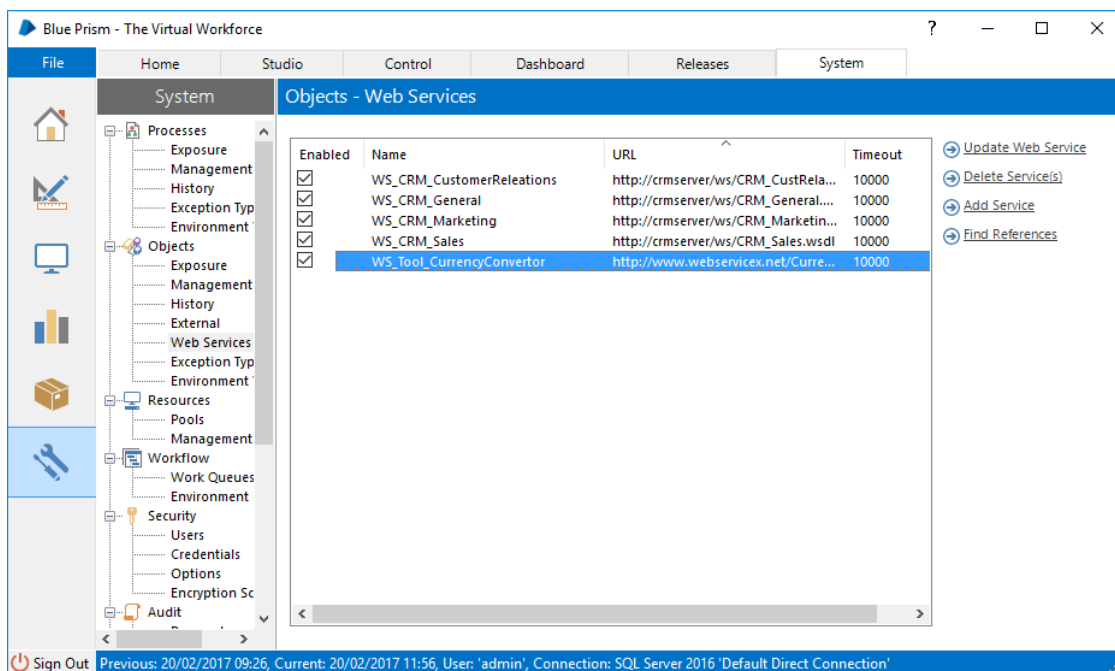
7. Select which **methods** from the web service should be available within Blue Prism and click **Next**.



- The wizard will suggest a name that will be used for the web service.
If the name is a duplicate of a previously exposed Blue Prism web service or if any invalid characters are detected a warning will be shown. Clicking Auto Correct will remove any invalid characters and ensure that the name is unique.



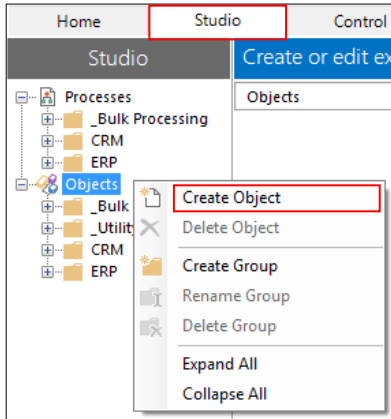
- The web service name will now appear in the list of Web Services that are enabled and available for use.



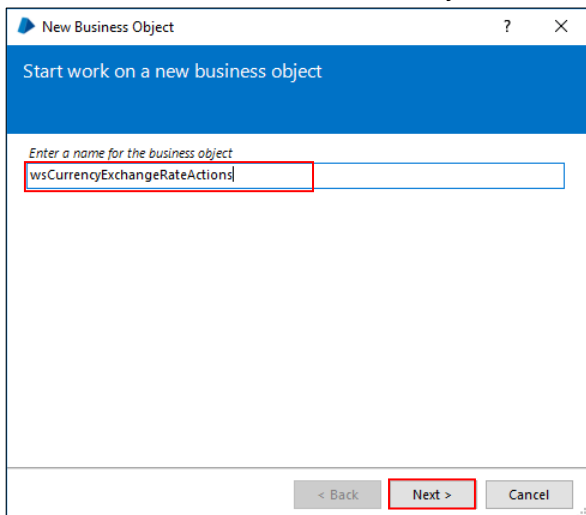
4.4. Walkthrough: Use a Third-Party Web Service in Blue Prism

This walkthrough provides an example of how to use a third-party web service which has been made available within Blue Prism. See the separate walkthrough for instructions on how to setup a third-party web service to be available in Blue Prism.

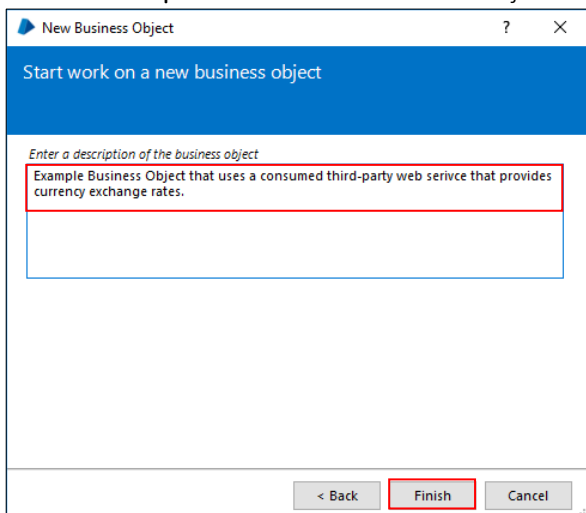
1. From the **Studio** tab, right-click and click **Create Object**.



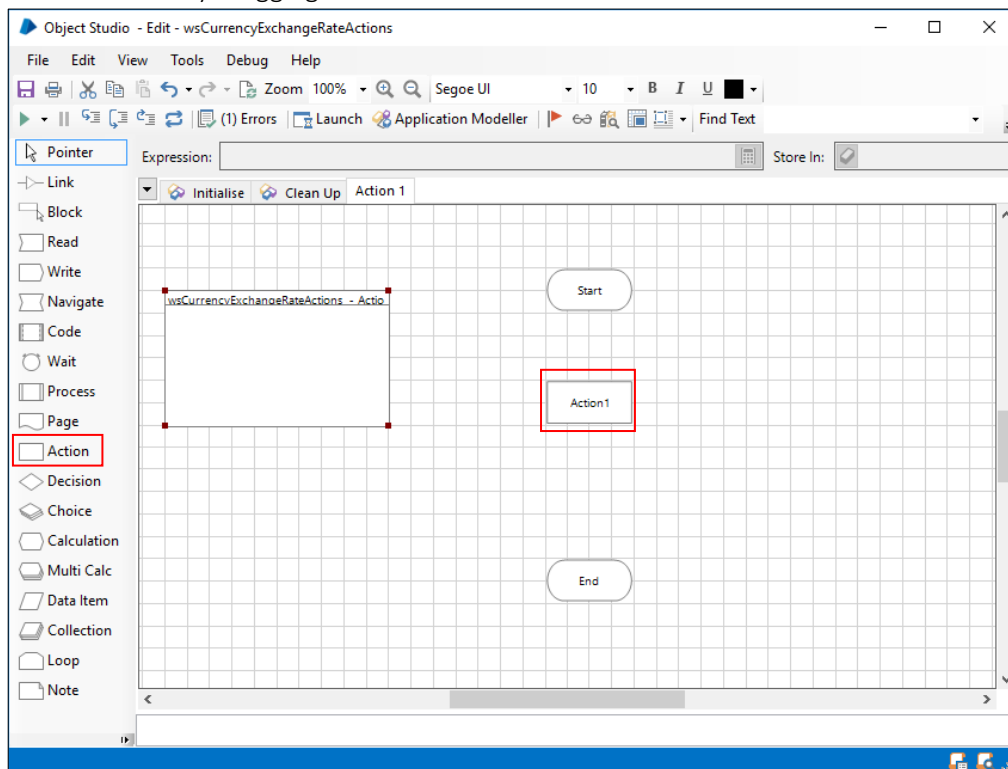
2. Enter a **name** for the new business object and click **Next**.



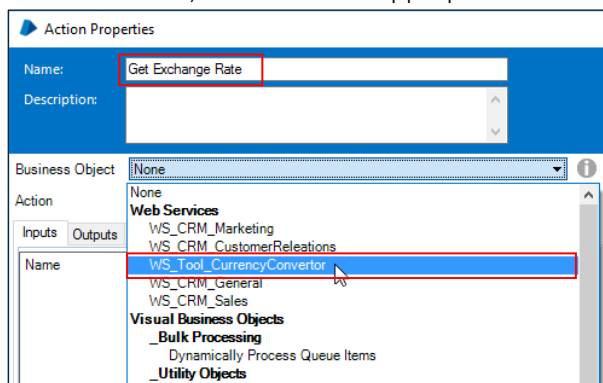
3. Enter a **description** for the new business object and click **Finish**.



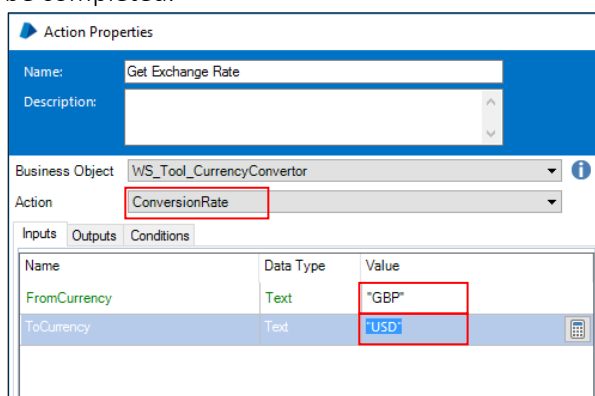
- Open the newly created Business Object by double-clicking, click on the page named **Action 1** and create an **Action** item by dragging the **Action** icon from the toolbar.



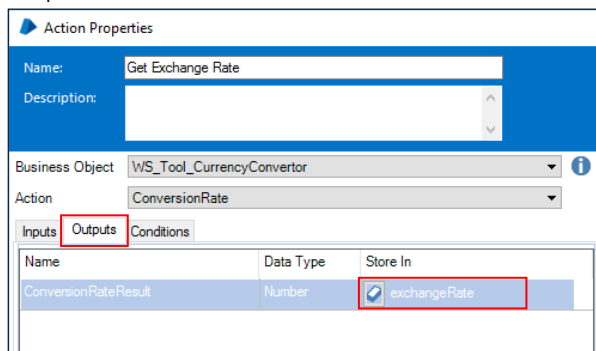
- Double click the **Action** item to set the properties. Enter the **Name**, and select the appropriate **web service** from the Business Object list.



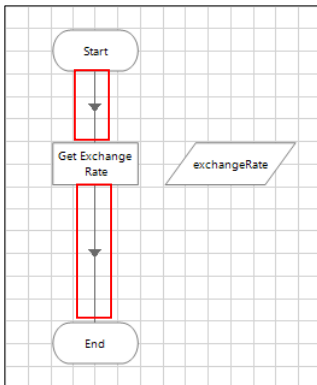
- Select the appropriate **action** from the list, note that any required **input parameters** are presented and can be completed.



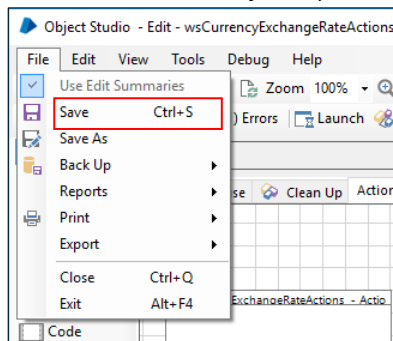
- Click the **Outputs** tab and define any necessary data items or collections that are required to store the output information. Click **OK**.



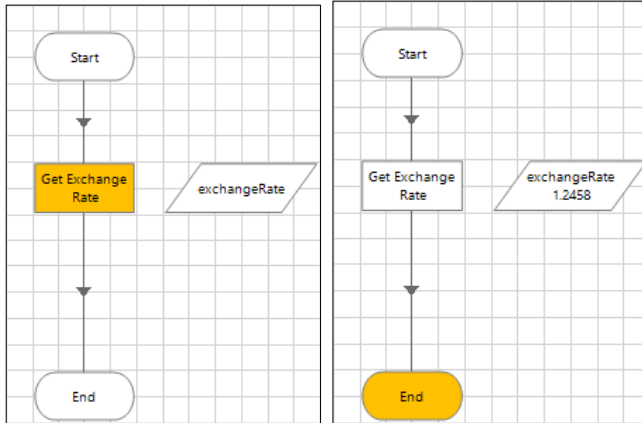
- Connect the nodes in the designer using the **Link** tool.



- Save the business object by clicking **File -> Save**.



10. Test the business object by clicking Reset and then using the Step button to progress through the nodes and review the value of the output parameters to confirm that the web service has responded successfully.



5. Advanced Topics

5.1. Design Considerations for a Blue Prism Web Service Interface

When deciding to expose Blue Prism web services for consumption by external systems there are a number of design considerations that should be reviewed to ensure that the approach both meets the intended requirement, and aligns with the capabilities of the Blue Prism platform.

- **Synchronous vs Asynchronous Processing**

If there is a requirement for requests to be fully serviced immediately upon receipt, it should be considered that the nature of processes often require interaction with various other systems and potentially complex rules and decision to be applied. This can result in the receiving Resource being busy for relatively long periods of time and, except for in a small number of scenarios, unable to accept additional requests during this period.

Due to the process complexity there may be significant merits to the overall solution if the processing can be handed-off to a “team” of Blue Prism Runtime Resources for asynchronous processing. This approach provides improved demand management by working items in order, as capacity becomes available. Additionally asynchronous processing can better leverage functionality provided by the platform such as message queuing, exception handling and reacting to variance in network and application performance.

- **Processing Volumes**

The pattern and volume of processing requests will also affect the design of the web service interface particularly if there is a requirement for concurrent requests to be received by a given Runtime Resource PC, or if there is the likelihood of large peaks in demand.

Consideration should be given to whether the Runtime Resources which receive the requests are able to receive multiple requests from a technical perspective. This is dictated by:

- Whether the nature of the request requires a new session to be created for each request received. This is further affected by the run mode of the sessions being created; as well as any other work that the Runtime Resource may be responsible for.
- Whether the platform licensing caters for the volumes of sessions that may be required at periods of maximum demand.

Subject to requirements, the web service interface can be designed such that both the requirement for new sessions to be created and the subsequent impact on the license can be both predicted and controlled.

Inappropriately designed interfaces could introduce availability and performance limitations, even when request volumes are low.

5.1.1. Session Management

Blue Prism sessions are a fundamental principle in way that Runtime Resources perform work and represent a runtime instance of an execution. They are also subject to license restrictions in that typically the Blue Prism license will limit the number of sessions which can run concurrently.

In order to understand whether the interface will be able to accept multiple concurrent requests both technically, and from a licensing perspective, it is important to understand how the design can impact session management and whether a single Resource PC can accept multiple concurrent connections.

5.1.1.1. Session Management Design

Sessions are managed differently depending on whether the web service request is directed at a Blue Prism Process or Business Object and whether auto-initialization is utilized.

- Connecting to an **Exposed Process**
Each request to an exposed Process will require a new session to be created resulting in potentially large numbers of sessions to be created which could impact licensing. It will also be necessary for the receiving Runtime Resource(s) to be able to have multiple concurrent active sessions. Typically this is limited by the run mode of any existing sessions as well as the run mode of any sessions being created.
- Connecting to a **Business Object using manual-initialization**
Each caller will programmatically manage the session creation resulting in the possibility that a single session could handle multiple requests. The likelihood is that a given Runtime Resource will need the ability to have multiple concurrent sessions active at a given time which could impact licensing and also requires the receiving Runtime Resources(s) to be able to have multiple concurrent active sessions.
- Connecting to a **Business Object using auto-initialization**
Blue Prism will manage the assignment of requests to a session automatically for processing requests to exposed Business Objects where the connection is set to use auto-initialization. This means that a single session per resource can potentially handle unlimited requests – the implication being that because new sessions are not continually required, the licensing impact is limited. The requirement for the Runtime Resource to have multiple concurrent active sessions is largely mitigated, unless the Resource is expected to concurrently handle other types of processing.

Whether or not an additional concurrent session can be created on a given Runtime Resource is based on:

- **Licensing**

Typically the Blue Prism license will limit the number of sessions which can be concurrently active across the estate and therefore there must be sufficient capacity available for the additional session to be created.

- **Session Run Mode**

The run mode of any current sessions on a given Runtime Resource as well as that of the newly requested session will impact whether or not it can be created. The run mode of the session is determined by the most restrictive setting applied to any of the Business Objects that are referenced by the Processes or Business Objects being executed.

The available Business Object run modes are listed below:

- **Foreground:** The object will never be permitted to have more than one active instance on a given Resource PC. This mode will allow the Business Object to be run at the same time as background Business Objects.

If the existing session is set to foreground any subsequent sessions which are also set to foreground or exclusive will be rejected until the original session has completed. Subsequent background sessions will be accepted

- **Background:** The object is designed to support multiple instances running concurrently on the same Resource PC. This mode will allow the Business Object to run at the same time as foreground and background business objects.

If the existing session is set to background any subsequent sessions which are set to exclusive will be rejected until the original session has completed. Subsequent background or foreground sessions will be accepted as long as there is no more than 1 foreground session present.

- **Exclusive:** This object will never be permitted to have more than one active instance on a given Resource PC and will not run at the same time as another Business Object.

If the existing session is set to exclusive any subsequent sessions will be rejected until the original session has completed. Likewise, irrespective of the run mode of any existing sessions, any subsequent requests for an exclusive session will be rejected until all previous sessions have been completed.

Consideration should also be given to any other sessions that may be generated on a given Runtime Resource. For example if a mixture of Processes and Business Objects are exposed, or if the Runtime Resource is also used to process ad-hoc or scheduled work, this will result in additional sessions being required.

5.1.1.2. Licensing and Session Management Examples

The following examples include a number of scenarios that illustrate session management and licensing implications.

Scenario 1

*“Runtime Resource 1” is currently executing a scheduled process “Process A” in **Foreground** run mode. In quick succession receives 25 web service requests targeted at “Process B” (which also operates in **Foreground** run mode).*

- a) A current active session is present for the execution of “Process A” (Foreground)
- b) “Runtime Resource 1” attempts to create a session for the first web service request however as it is not possible to create a Foreground session when one of the same type is ready present. The request is rejected with a message indicating that the Runtime Resource is busy.
- c) “Runtime Resource 1” attempts to create a session for each of the subsequent web service requests. As these requests have arrived prior to the original Foreground session request being completed, and the request being for Foreground processing, these requests will also be rejected with a message indicating that the Runtime Resource is busy.

Total Sessions: 1

Rejected Requests: 25 (excluding rejections due to licensing capacity)

Scenario 2

*“Runtime Resource 1” is currently executing a scheduled process “Process A” in **Foreground** run mode. In quick succession receives 25 web service requests targeted at “Process C” (which operates in **Background** run mode).*

- a) A current active session is present for the execution of “Process A” (Foreground)
- b) “Runtime Resource 1” attempts to create a session for the first web service request, the session is created successfully assuming that there is sufficient licensing capacity.
- c) “Runtime Resource 1” attempts to create a session for each of the subsequent web service requests. Subject to sufficient licensing capacity each of these sessions will be successfully created.

Total Sessions: 26

Rejected Requests: None (excluding rejections due to licensing capacity)

Scenario 3

*“Runtime Resource 1” is currently executing a scheduled process “Process D” in **Exclusive** run mode. In quick succession receives 25 web service requests targeted at “Process C” (which operates in **Background** run mode).*

- a) A current active session is present for the execution of “Process D” (Background)
- b) “Runtime Resource 1” attempts to create a session for the first web service request, however this is rejected as the existing session is set to run in Exclusive mode.
- c) “Runtime Resource 1” attempts to create a session for each of the subsequent web service requests. These are all rejected as the existing session is set to run in Exclusive mode.

Total Sessions: 1

Rejected Requests: 25 (excluding rejections due to licensing capacity)

Scenario 4

Runtime Resources (1, 2 & 3) are configured with two **Business Objects** (A & B).

“Business Object A” requires a **Foreground** run mode, whereas “Business Object B” requires an exclusive run mode.

In quick succession, Runtime Resources 1 and 2 receive 25 web service requests targeted at “Business Object A”, and “Runtime Resource 3” receives 10 requests targeted at “Business Object A” and 10 requests targeted at “Business Object B”. The requests specify that **auto-initialization** should be used.

- a) “Runtime Resource 1” attempts to create and start a session. It succeeds because there isn’t already an active Exclusive run mode session on this Runtime Resource and because there is licensing capacity. The single session processes all 25 requests for “Business Object A” (Foreground).
- b) “Runtime Resource 2” attempts to create and start a session. It succeeds because there isn’t already an active session, irrespective of run mode, on this Runtime Resource and because there is licensing capacity. The single session processes all 25 requests for “Business Object B” (Exclusive). Auto-initialisation allows the Business Objects in this scenario to have any run mode.
- c) “Runtime Resource 3” attempts to create and start a session. It succeeds because there isn’t already an active session, irrespective of run mode, on this Runtime Resource and because there is licensing capacity. The single session processes 10 requests for “Business Object A” (Foreground) followed by 10 requests for “Business Object B” (Exclusive).

Total Sessions: 3

Rejected Requests: None (excluding rejections due to licensing capacity)

In this example use of auto-initialization when calling the Business Object web services allows for a mixture of run modes (including exclusive) to be concurrently received for processing by a single Runtime Resource.

Scenario 5

Consider a scenario whereby there are 3 Runtime Resources (1, 2 & 3) that are configured with two Processes (E & F). "Process E" runs in a **Background** mode, whilst "Process F" requires an **Exclusive** run mode.

In quick succession, Runtime Resources 1 and 2 each receive 25 web service requests targeted at "Process E", and "Runtime Resource 3" receives 10 requests targeted at "Process E" and 10 Requests targeted at "Process F".

- a) "Runtime Resource 1" attempts to create and start a session for each of the 25 requests. It succeeds as long there isn't already an active Exclusive run mode session on this Runtime Resource and if there is licensing capacity.
- b) "Runtime Resource 2" attempts to create and start a session for each of the 25 requests. It succeeds as long there isn't already an active Exclusive run mode session on this Runtime Resource and if there is licensing capacity.
- c) "Runtime Resource 3" attempts to create and start a session for each of the 10 requests. It succeeds as long there isn't already an active Exclusive run mode session on this Runtime Resource and if there is licensing capacity.
- d) "Runtime Resource 3" attempts to create and start a session for the first of the 10 requests for "Process F" however this is rejected because Runtime Resource 3 cannot allow an Exclusive run mode session due to there being other active sessions already present.

Total Sessions: 60

Rejected Requests: 10 (excluding rejections due to licensing capacity)

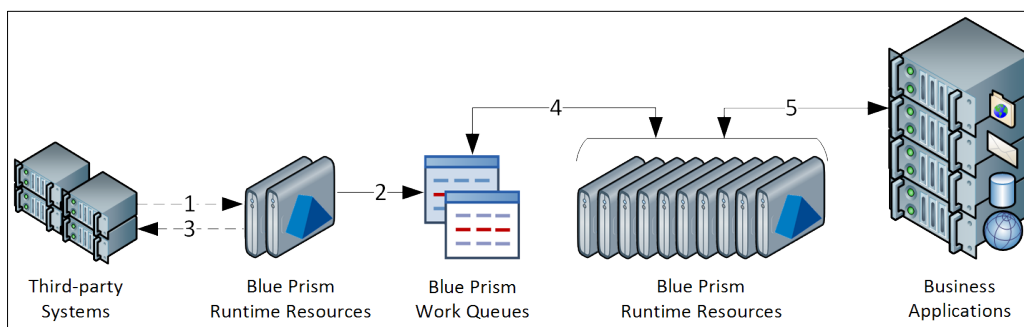
Blue Prism Recommended Approach

Whilst there are potentially many approaches which may be suited to a range of requirements, the most common Blue Prism web service interface design includes the following features:

- Expose only Blue Prism Business Object(s), which simply verify and the pass data into a specified work queue. Ideally these Business Objects will be set to run in Background mode however scenarios featuring foreground and exclusive run modes can potentially be supported with this configuration. In some cases it may be appropriate to have a single object action with parameters that allow the caller to specify the data and the target work queue. Other scenarios may include exposing a Business Object with an action for each of the target work queues.
- Allocate a small number of dedicated Blue Prism Runtime Resources to receive the web service requests. The caller uses auto-initialization in order to minimise the number of Blue Prism sessions that will be initiated and to cater for a mixture of run modes to be used within a single session if required. Where multiple sessions are required on a single Runtime Resources such as where it is used for other processing, the run modes will need to be configured such that they do not conflict.
- Configure a Blue Prism Process to work the items in the Work Queue. (This can be as comprehensive as required)
- The control of the subsequent processing of the work items is aligned to the other work queues managed by the Blue Prism platform. (E.g. it is managed automatically using schedules, or manually by the Blue Prism controllers). This allows the number of Runtime Resources dedicated to clearing down the work queue can be tailored to meet demand.
- The output of the process is handled by Blue Prism and can include a wide range of options. A non-exhaustive list of options includes: updating a target system, updating a database, logging the result into a flat file, sending an email, calling a third-party web service etc.

5.1.1.3. Example

The diagram presents a common configuration when exposing Blue Prism web services.



1. The web service request is received by the Runtime Resource PC. The request targets a Blue Prism Business Object and contains information about the item that requires processing.
2. Based on the action called and the data passed, the information is verified and a work item is added to the appropriate Blue Prism Work Queue.
3. The web service returns a response that indicates whether adding the item to work queue was successful.
4. Based on the automatic schedules, the Blue Prism Runtime Resources start a Process to query the work queue and collaboratively work on each item in turn. (Can also be triggered manually by a Blue Prism controller)
5. Where required as part of the Process execution, the Blue Prism Runtime Resources interact with various business applications.

5.1.1.4. Advantages

The main advantages of the Blue Prism recommended approach are:

- The low-complexity of the Business Object actions allows the run mode to be set to background operation which enables simultaneous requests to be received.
- Caters for high variance in demand of the web service whilst minimising the number of Blue Prism sessions that will be required. This allows the licensing impact to be predicted and managed.
- Segregation of the logging of the work queue item from the subsequent processing allows for the capacity of the processing to be scaled independently of the capacity to receive requests. Furthermore the asynchronous nature of the subsequent processing allows for the number of Runtime Resources assigned to the work queue to be scaled in line with demand and can leverage complex decisioning and comprehensive exception handling capabilities.

5.2. Securing Exposed Blue Prism Web Services

Where there is a requirement to provide Blue Prism web service connectivity to systems which are outside of the trusted, secure local network, it may be desirable for the communication to take place over HTTPS using Secure Socket Layers (SSL). There are two common approaches for this:

- Configure Blue Prism Runtime Resource listeners to use certificate-based encryption for incoming instructional communication.
- Deploy a proxy server to implement SSL. This would inline between the external requestor and the Blue Prism Runtime Resources.

5.2.1. Configuring Blue Prism Runtime Resources to use certificate-based encryption

For Blue Prism 5.0.18 and above, Runtime Resources can be configured to use a local certificate to apply certificate-based encryption for all traffic received on the Resource listening port (default 8181).

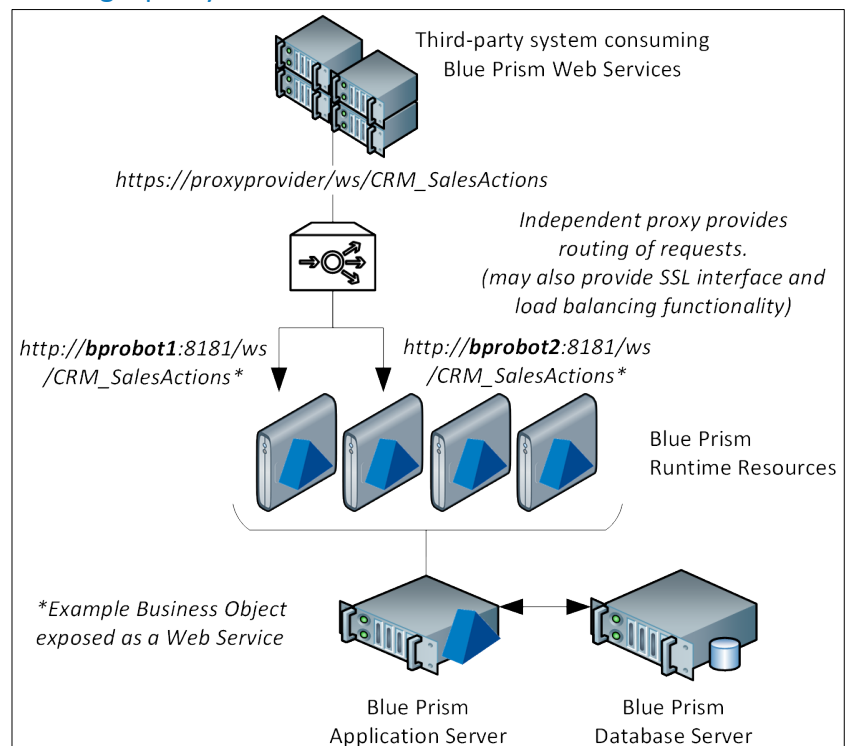
This affects all instructional information received by the Runtime including instructions from Blue Prism Interactive Clients, Blue Prism Server (scheduler) and Web Service queries. The URL when consuming web services hosted on a secured Blue Prism Runtime Resource will use a HTTPS prefix (e.g. <https://bprobot1:8181>)

Further information is provided within the **Blue Prism Data Sheet – Securing Network Connectivity**

5.2.2. Securing Blue Prism Web Services using a proxy

In order to achieve this configuration, one option is to implement a proxy server as shown in the diagram. Dependent on the proxy server that is implemented, there may also be scope to leverage additional features such as load balancing which can distribute the communications between the online available Runtime Resources.

In this scenario the proxy would only be used to marshal for external (non-Blue Prism) traffic.



5.3. Consume Web Services using Code Stage Based Business Objects

Some web services contain features that are not natively compatible with Blue Prism. In these instances code stages can be used to build a custom web service wrapper to allow interpretation and use by Blue Prism.

This guide describes how to build a web service wrapper for such scenarios based on code stages and uses Microsoft Web Service Integration technologies.

These technologies are mature and support features such as polymorphic-typed web service action parameters through use of auto-generated code. It should be noted however that such auto-generated code is static and changes to the web service may require re-work.

Additional steps are included within the example to provide instructions on how the code stage can be used to pass credentials in the SOAP header for web services which utilize WS-Security authentication, implemented by WSE 3.0. The alternative or additional steps within the example are marked as **WS-Security Implementation**.

Code stages can be further customized to control additional custom features such as toggling the **MustUnderstand** attribute, or defining if the WS-Addressing header should be included however this is outside the scope of this example

5.3.1.1. Pre-requisites

This guide is intended for technical users with the following skills:

- A basic working knowledge of web services. In particular it will be useful to be familiar with WSDL documents and if relevant, WS-Security.
- Good knowledge of the Blue Prism product including how to create objects, actions and stages within those objects.
- Good understanding of writing applications in Visual Studio .NET using either C# or Visual Basic.

The following tools will be required:

- A working installation of Blue Prism with appropriate user account privileges to create, modify, publish and run processes.
- Visual Studio 2005 or above.

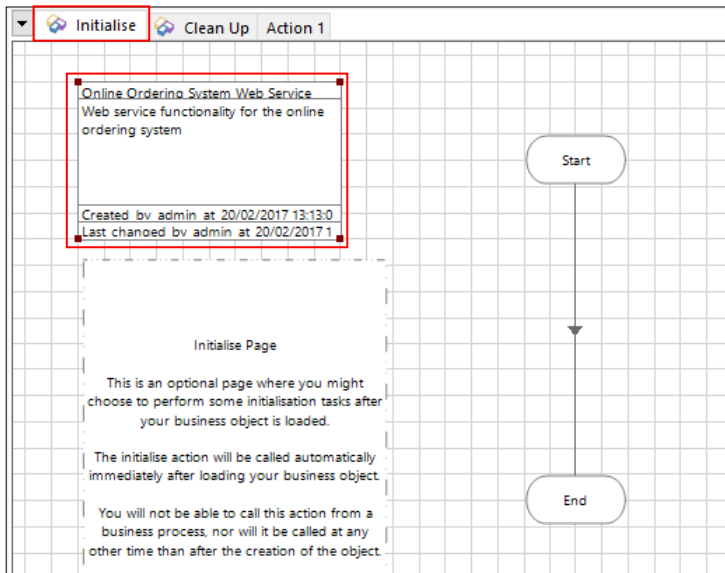
5.3.1.2. Steps

The following steps are addressed by this guide:

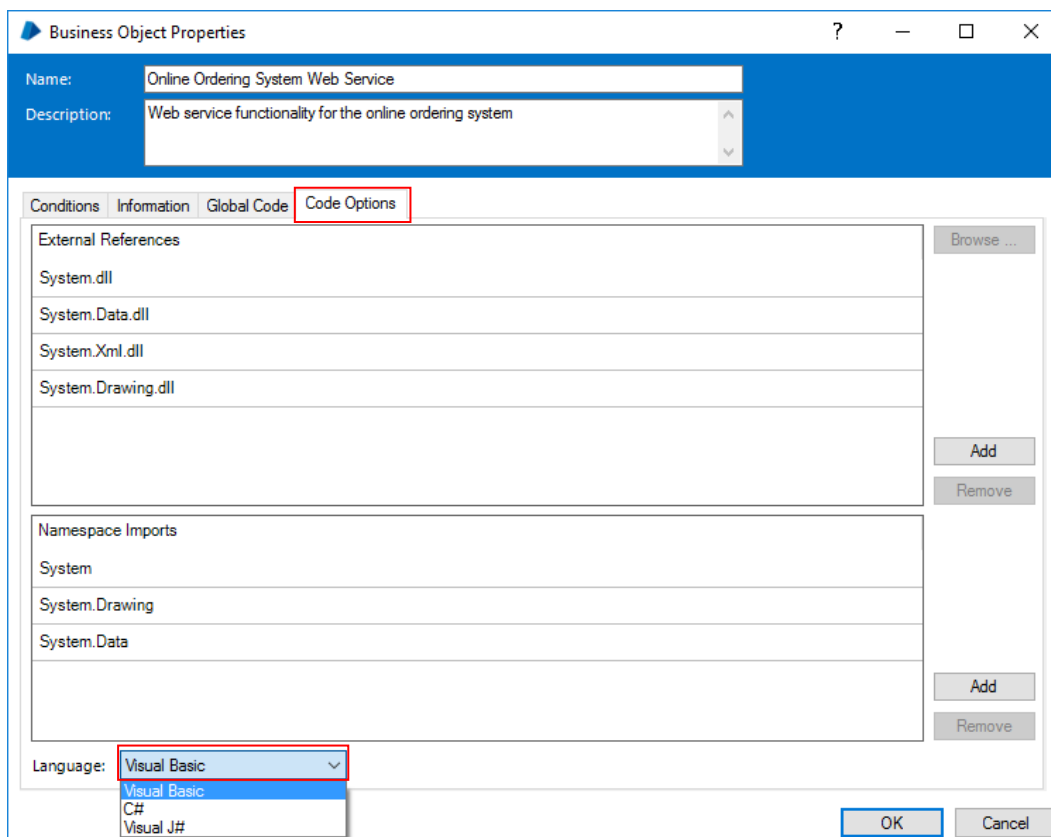
1. Generate a new Business Object that will be configured to provide the interface between Blue Prism and the web service and select the preferred coding language.
2. Use the appropriate tool to auto-generate the base code for interpreting the WSDL.
 - a. If following the **WS-Security Implementation** example, it is necessary to install WSE on each of the developer Interactive Clients and each Runtime Resource PC
3. Set up a Blue Prism Code Stage with the appropriate elements of the generated code, and the necessary reference libraries.
4. Define the web service URL as an environment variable for ease of updating it in the future.
5. Generate a web service wrapper to transform the inputs and outputs between Blue Prism and the web service.

5.3.2. Generate a new Business Object

1. Create a new **business object** within Blue Prism. Give it a name and description.
2. Once the object has been created click on the tab which says **Initialise**, and then **double click** on the **Page Info** stage (pictured below).



3. Note that this is where the **Global Code** and **Code Options** tabs are located. Click **Code Options** and select the preferred language.



5.3.3. Auto-Generate Code using wsdl.exe or wsewsdl3.exe

Typically the Windows SDK is installed as part of the installation of Visual Studio and this contains the **wsdl.exe** tool which can be used to create a wrapper class around the web service.

Alternatively when WSE 3.0 is installed, the alternative **wsewsdl3.exe** tool can be used.

This should be used when consuming a web service which implements WS-Security and requires credentials to be passed in the SOAP header.

The wrapper class that is generated is then used within the Global Code of the Business Object.

1. Launch the **Visual Studio Command Prompt**

Typically found from the relevant **Microsoft Visual Studio** folder within the **Start Menu**.

(E.g. Programs -> Microsoft Visual Studio -> Visual Studio Tools -> Visual Studio Command Prompt).

2. Use either the wsdl.exe or wsewsdl.exe tool via the command prompt against the web service WSDL **wsdl.exe**

Suitable for web services which do not implement WS-Security and do not require credentials to be passed in the SOAP header

```
wsdl.exe /language:[VB|CS] /out:[File Location] [Address of WSDL]
```

VB .NET

```
wsdl.exe /language:VB /out:"%TEMP%" http://example.com/wsService?wsdl
```

C#

```
wsdl.exe /language:CS /out:"%TEMP%" http://example.com/wsService?wsdl
```

WS-Security Implementation:

When following the WS-Security additional example the wsewsdl3.exe tool should be used.

wsewsdl3.exe

Suitable for web services which implement WS-Security and require credentials to be passed in the SOAP header

```
wsewsdl3.exe /language:[VB|CS] /type:webClient /out:[File Location] [Address of WSDL]
```

VB .NET

```
wsewsdl3.exe /language:VB /type:webClient  
/out:"%TEMP%" http://example.com/wsService?wsdl
```

C#

```
wsewsdl3.exe /language:CS /type:webClient  
/out:"%TEMP%" http://example.com/wsService?wsdl
```

3. The output of the tool should look similar to the message below and indicates the specific location of the output file that has been generated:

```
Microsoft (R) Web Services Description Language Utility  
[Microsoft (R) .NET Framework, Version 2.0.50727.42]  
Copyright (C) Microsoft Corporation. All rights reserved.  
Writing file 'C:\Temp\Service1.vb'.
```

4. Navigate to the file location and open it using a suitable editor.

5.3.4. Copy the relevant code to the Business Object

The code generated by the wsdl.exe tool contains a header that must be removed prior to the remainder of the code being copied to the previously created Business Object.

The header that can be discarded is similar to below:

VB .NET

```
'-----
' <auto-generated>
'   This code was generated by a tool.
'   Runtime Version:2.0.50727.5466
'
'   Changes to this file may cause incorrect behavior and will be lost if
'   the code is regenerated.
' </auto-generated>
'-----

Option Strict Off
Option Explicit On

Imports System
Imports System.ComponentModel
Imports System.Diagnostics
Imports System.Web.Services
Imports System.Web.Services.Protocols
Imports System.Xml.Serialization
'
'This source code was auto-generated by wsdl, Version=2.0.50727.42.
'
'''<remarks/>
```

C#

```
//-----
// <auto-generated>
//   This code was generated by a tool.
//   Runtime Version:2.0.50727.5466
//
//   Changes to this file may cause incorrect behavior and will be lost if
//   the code is regenerated.
// </auto-generated>
//-----

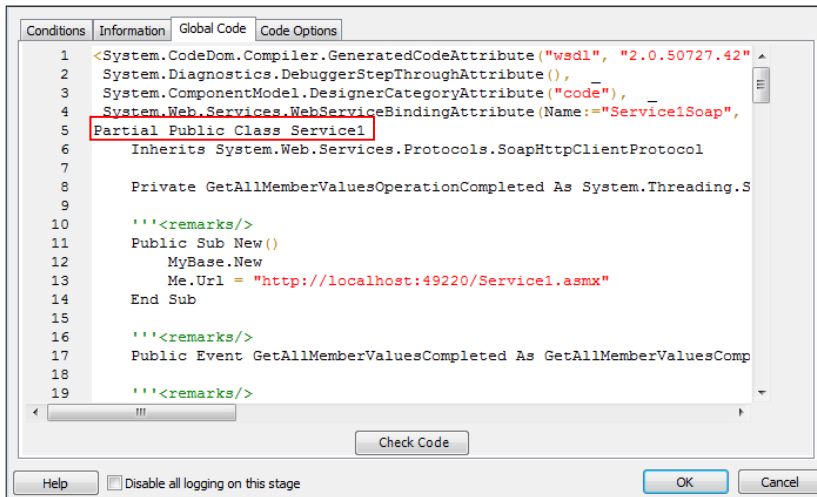
using System.ComponentModel;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Xml.Serialization;

//
//This source code was auto-generated by wsdl, Version=2.0.50727.42.
//
```

Manually copy the remaining code into the **Global Code** tab of the **Business Object**. Do not press the **Check Code** button at this time.

Make a note of the name of the class (in the code examples below it is **Service1**).

VB . NET



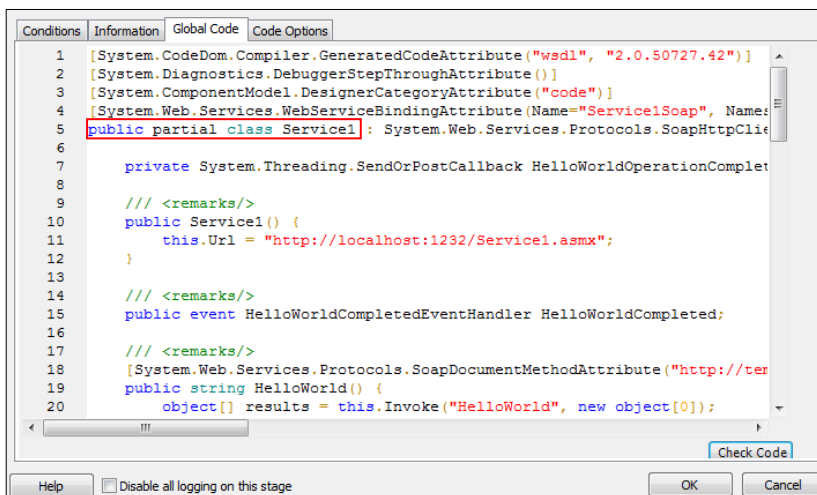
```

1 <System.CodeDom.Compiler.GeneratedCodeAttribute("wsdl", "2.0.50727.42")
2 System.Diagnostics.DebuggerStepThroughAttribute(),
3 System.ComponentModel.DesignerCategoryAttribute("code"),
4 System.Web.Services.WebServiceBindingAttribute(Name="Service1Soap",
5 Partial Public Class Service1
6 Inherits System.Web.Services.Protocols.SoapHttpClientProtocol
7
8 Private GetAllMemberValuesOperationCompleted As System.Threading.S
9
10 '''<remarks/>
11 Public Sub New()
12 MyBase.New
13 Me.Url = "http://localhost:49220/Service1.asmx"
14 End Sub
15
16 '''<remarks/>
17 Public Event GetAllMemberValuesCompleted As GetAllMemberValuesComp
18
19 '''<remarks/>

```

Buttons: Help, Disable all logging on this stage, Check Code, OK, Cancel

C#



```

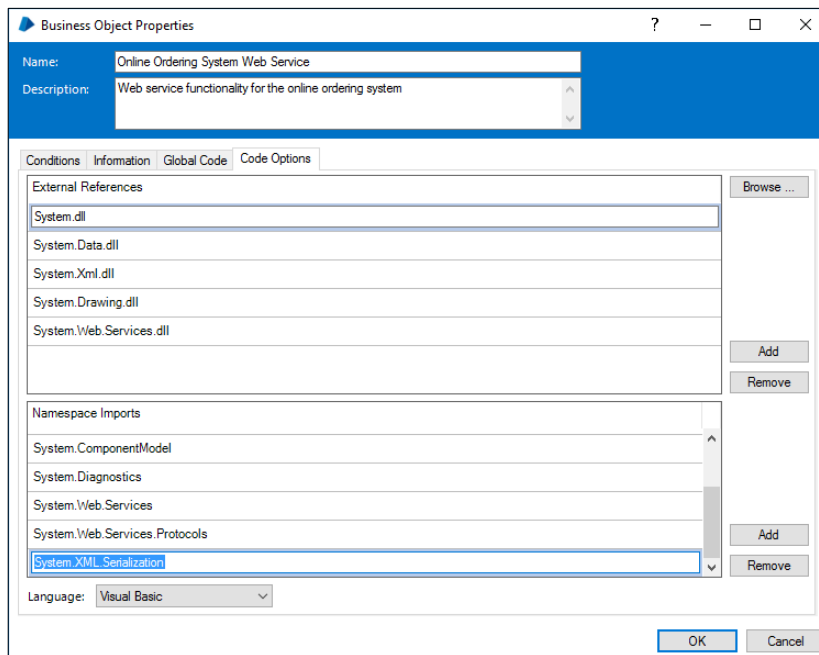
1 [System.CodeDom.Compiler.GeneratedCodeAttribute("wsdl", "2.0.50727.42")]
2 [System.Diagnostics.DebuggerStepThroughAttribute()]
3 [System.ComponentModel.DesignerCategoryAttribute("code")]
4 [System.Web.Services.WebServiceBindingAttribute(Name="Service1Soap", Name=
5 public partial class Service1 : System.Web.Services.Protocols.SoapHttpClie
6
7 private System.Threading.SendOrPostCallback HelloWorldOperationComple
8
9 /// <remarks/>
10 public Service1() {
11 this.Url = "http://localhost:1232/Service1.asmx";
12 }
13
14 /// <remarks/>
15 public event HelloWorldCompletedEventHandler HelloWorldCompleted;
16
17 /// <remarks/>
18 [System.Web.Services.Protocols.SoapDocumentMethodAttribute("http://ter
19 public string HelloWorld() {
20 object[] results = this.Invoke("HelloWorld", new object[0]);

```

Buttons: Help, Disable all logging on this stage, Check Code, OK, Cancel

5.3.5. Add the necessary External Library References

1. Click into the Code Options tab to add the following External References and Namespace Imports:



VB .NET

```
External References
System.Web.Services.dll

Namespace Imports
System.ComponentModel
System.Diagnostics
System.Web.Services
System.Web.Services.Protocols
System.Xml.Serialization
```

C#

```
External References
System.dll
System.Web.Services.dll

Namespace Imports
System.ComponentModel
System.Diagnostics
System.Web.Services
System.Web.Services.Protocols
System.Xml.Serialization
```

WS-Security Implementation

When following the WS-Security the following additional Code Options are required:

```
External References
Microsoft.Web.Services3.dll

Namespace Imports
Microsoft.Web.Services3.Design
Microsoft.Web.Services3.Messaging
Microsoft.Web.Services3
Microsoft.Web.Services3.Security.Tokens
```

2. Select the **Global Code** tab and click **Check Code**. If any errors are reported, these must be resolved before continuing.

5.3.6. Setup the Global Code to use an Environment Variable for the URL

To allow future modification of the web service URL, the constructor of the **Service1** class is going to be modified to use the value specified in a Blue Prism Environment Variable.

1. Within the **Global Code** of the Business Object, at the top of the existing code, add a member variable:

```
VB.NET
Private mWebService As Service1
```

```
C#
private Service1 mWebService;
```

2. Locate the **Service1** class **constructor** within the code and add a URL string parameter. Typically located by searching for

```
VB.NET
Public Sub New ()
```

```
C#
public Service1()
```

The modified constructor should resemble the following:

```
VB.NET
Public Sub New(ByVal url As String)
    MyBase.New()
    Me.Url = url
End Sub
```

```
C#
public Service1(string url) : base()
{
    this.Url = url;
}
```

3. Create a new environment variable using the Blue Prism System Manager (Process Area -> Environment Variables). Click the **add variable** link and give the environment variable a name, a type of text, and the value of the URL as it appeared in the code originally.
4. Within the initialize page of the Business Object create a data item that uses the newly created environment variable. The name of the data item must exactly match the name of the environment variable. Choose environment variable from the exposure type drop down within the data item properties.
5. Also within the Initialise page, add a code stage with a single input called 'Url' of type text, which will be populated using a Blue Prism environment variable. The code to be added to this code stage is:

```
VB.NET
mWebService = New Service1(Url)
```

```
C#
mWebService = new Service1(Url);
```

The code stage should be linked to the start and end stages of the initialize page.

5.3.7. Generate a Wrapper Code Stage

In order to allow Blue Prism processes to use the web service a code stage wrapper is required. This will involve creating a new **Action** within the **Business Object**.

Writing wrapper code stages requires bespoke implementation and is dependent on the action of the web service that needs to be invoked.

This part of the guide provides an overview of what needs to be achieved, but the examples will need to be modified to meet the specific requirements of the web service action.

The actions required to generate the wrapper code stage include:

1. Create a new Action page within the Business Object
2. Define the appropriate input and output parameters for the object Action based on those required for the web service action.

Blue Prism presents data types to code stages as native .NET data types. E.g.:

- o *Text* is presented to the code stage as a *System.String*
 - o *Number* is presented to the code stage as *System.Decimal*.
 - o A *Collection* datatype is presented to the code stage as a *System.Data.DataTable*. A single row collection can be used to represent name value pairs and this could then be mapped to the member properties of a class. A collection with a single field can be used to represent array or list data. A nested collection can be used to represent hierarchical or tree like data structures.
3. Add a code stage to the action and link it to the start and end stages. Input and Output parameters for the code stage should then be set mirroring those set as inputs and outputs of the action.
 4. Within the code stage itself as a basic outline it is expected that the code does at least the following:
 - a. Convert input parameters to web service action parameters
 - b. Call the web service action
 - c. Convert the result of the web service call to output parameters.
 5. The following should be noted about the following example:
 - a. That a loop has been implemented to process the incoming collection to ensure that all rows are accounted for. It is recommended to read collections in this way even if it is expect that the collection will contain a single row.
 - b. The columns specified for the output data table must match the field names defined in the collection within the Blue Prism object.

An example of the code that is required is below. This code is provided as an example and is not expected to compile without being modified.

The indented code encapsulated with purple comments is only required when following the **WS-Security Implementation** example.

VB.NET

```
Dim q As New QueryByName() 'Replace QueryByName with the correct class name
'Example 1 Collection input
Dim inputDataTable As System.Data.DataTable = input1
For Each inRow As System.Data.DataRow In inputDataTable.Rows
    q.Value = inRow("field1")
Next
'Example 2 Number Input
Dim inputNumber As Integer = input2
q.NumberValue = inputNumber
'Example 3 Text Input
Dim inputText As String = input3
q.TextValue = inputNumber
'...

'Example of Invoking the web service action
Dim r As BaseResponse
```

```
'Content required for WS-Security Implement: Start
Dim token As New UsernameToken("webSvcUser", "Pa55word", PasswordOption.SendPlainText)
mWebService.SetClientCredential(Of UsernameToken)(token)
Dim webServiceClientPolicy As New Policy()
webServiceClientPolicy.Assertions.Add(New UsernameOverTransportAssertion())
mWebService.SetPolicy(webServiceClientPolicy)
'Content required for WS-Security Implement: END
```

```
r = mWebService.retrievevaluationsBy(q)

'Example of converting the output of the web
'service action into a datatable for storing
'in the output collection
Dim outputDataTable As New System.Data.DataTable
outputDataTable.Columns.Add("field1")
Dim outRow As System.Data.DataRow = outputDataTable.Rows.Add()
outRow("field1") = r.Value

output1 = outputDataTable
'...
```

c#

```
QueryByName q = new QueryByName(); //Replace QueryByName with correct class name

//Example 1 Collection input
System.Data.DataTable inputDataTable = input1;
foreach (System.Data.DataRow inRow in inputDataTable.Rows) {
    q.Value = inRow["field1"];
}

//Example 2 Number Input
int inputNumber = input2;
q.NumberValue = inputNumber;

//Example 3 Text Input
string inputText = input3;
q.TextValue = inputNumber;
//...

//Example of Invoking the web service action
BaseResponse r = default(BaseResponse);

//Content required for WS-Security Implement: Start
UsernameToken token = new UsernameToken("webSvcUser", "Pa55word", PasswordOption.SendPlainText);
mWebService.SetClientCredential<UsernameToken>(token);
Policy webServiceClientPolicy = new Policy();
webServiceClientPolicy.Assertions.Add(new UsernameOverTransportAssertion());
mWebService.SetPolicy(webServiceClientPolicy);
//Content required for WS-Security Implement: END
r = mWebService.retrieveEvaluationsBy(q);

//Example of converting the output of the web
//service action into a datatable for storing
//in the output collection
System.Data.DataTable outputDataTable = new System.Data.DataTable();
outputDataTable.Columns.Add("field1");
System.Data.DataRow outRow = outputDataTable.Rows.Add();
outRow["field1"] = r.Value;
output1 = outputDataTable;
//...
```

5.4. Consume an Exposed Blue Prism Web Service using Visual Studio

This guide illustrates how to build an independent web service client using Visual Studio which can consume a web service that has been exposed by Blue Prism.

It is a full working example which includes use of a provided example components.

5.4.1.1. Pre-requisites

This guide is intended for technical users with the following skills:

- A basic working knowledge of web services. In particular it will be useful to be familiar with WSDL documents.
- Good knowledge of the Blue Prism product including how to expose a Business Objects and/or Process as web services.
- Good understanding of working with Visual Studio using either C# or Visual Basic .NET. This example uses **VB .NET**.

The following tools will be required:

- A working installation of Blue Prism with appropriate user account privileges to create, modify, publish and run processes.
- The sample Blue Prism Business Object and Process.
- Visual Studio 2005 or above.

5.4.1.2. Steps

The following steps are addressed by this guide:

1. Download and Import the Sample Business Object and Process
2. Exposing the Business Object and Process as web services
3. Create a new Visual Studio Console Application Project with the appropriate library references.
4. Add a web reference linking to the exposed web service.
5. Produce simple code to:
 - a. Interact with a Business Object based web service (using manual and auto-initialization).
 - b. Interact with a Process based web service.
 - c. Trigger the above interactions when the application runs.

5.4.2. Download and Import the Sample Business Object and Process

A resource zip file is available via the user portal which contains:

- A sample Blue Prism Business Object
- A sample Blue Prism Process
- Example Visual Studio Project (VB.NET) of a Web Service Client

Search on the portal for: **Resources_Consuming an Exposed Blue Prism Web Service.zip**

1. Launch **Blue Prism** and Log in
2. From the **File** menu, use the **Import** wizard to individually import the sample **Business Object** and **Process**.
 - a. BPA Object – NumericOperations.xml
 - b. BPA Process – FavouriteFruitLookup.xml

5.4.3. Expose the Sample Business Object and Process as Web Services

1. Use the wizards located accessed via the following locations to expose the Business Object and Process as web services. For the purposes of this example names of the respective exposed web services should match those stated below:

	Wizard Location	Web Service Name
Business Object	System Manager -> Business Objects -> Process Exposure	NumericOperations
Process	System Manager -> Processes -> Process Exposure	FavouriteFruitLookup

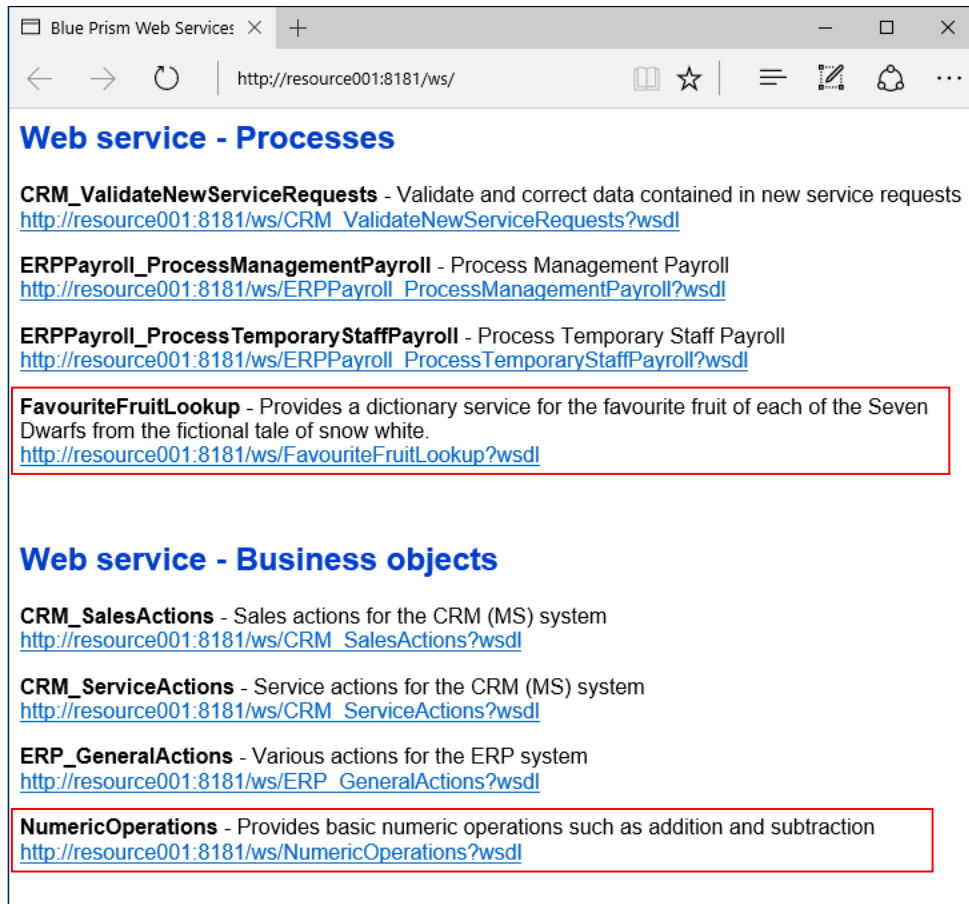
Full walkthroughs on how to: Expose a Business Object as a Web Service; and Expose a Process as a Web Service are available.

2. Retrieve the **URL** of the **WSDL** for the respective web services for use later on by browsing to the following URL:
[http://\[machine name\]:\[port\]/ws/](http://[machine name]:[port]/ws/)

Where **[machine name]** is the name of one of the Blue Prism runtime resources, and **[port]** is the defined port that Blue Prism is configured for.

E.g. <http://resource001:8181/ws/>

Where the Blue Prism Runtime Resource has been configured with certificate-based encryption, the prefix will be **https**.



Blue Prism Web Services

http://resource001:8181/ws/

Web service - Processes

CRM_ValidateNewServiceRequests - Validate and correct data contained in new service requests
http://resource001:8181/ws/CRM_ValidateNewServiceRequests?wsdl

ERPPayroll_ProcessManagementPayroll - Process Management Payroll
http://resource001:8181/ws/ERPPayroll_ProcessManagementPayroll?wsdl

ERPPayroll_ProcessTemporaryStaffPayroll - Process Temporary Staff Payroll
http://resource001:8181/ws/ERPPayroll_ProcessTemporaryStaffPayroll?wsdl

FavouriteFruitLookup - Provides a dictionary service for the favourite fruit of each of the Seven Dwarfs from the fictional tale of snow white.
<http://resource001:8181/ws/FavouriteFruitLookup?wsdl>

Web service - Business objects

CRM_SalesActions - Sales actions for the CRM (MS) system
http://resource001:8181/ws/CRM_SalesActions?wsdl

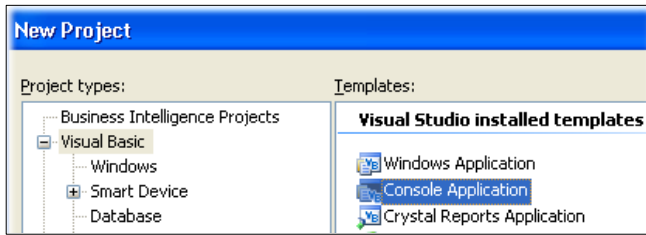
CRM_ServiceActions - Service actions for the CRM (MS) system
http://resource001:8181/ws/CRM_ServiceActions?wsdl

ERP_GeneralActions - Various actions for the ERP system
http://resource001:8181/ws/ERP_GeneralActions?wsdl

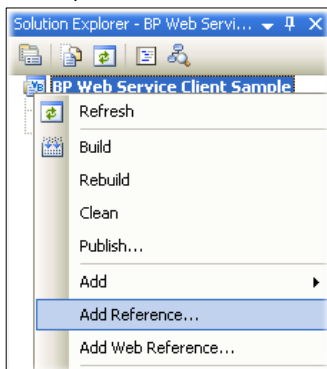
NumericOperations - Provides basic numeric operations such as addition and subtraction
<http://resource001:8181/ws/NumericOperations?wsdl>

5.4.4. Use Visual Studio to create a new Console Application

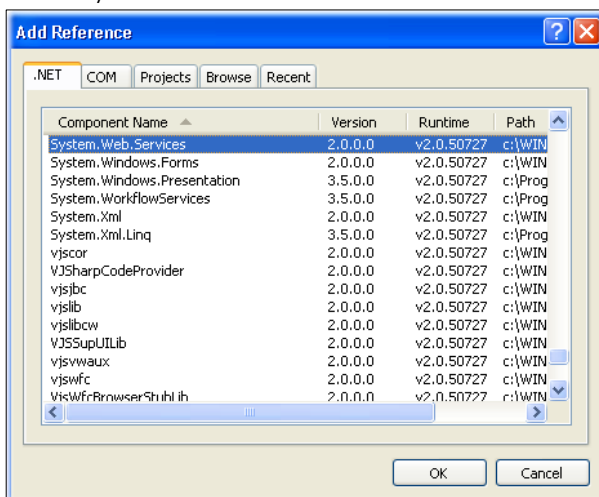
1. Create a new Visual Basic Console Application, named BP Web Service Client Sample



2. Right-click the project name within the **solution explorer** and select **Add Reference** to add the required Library References.

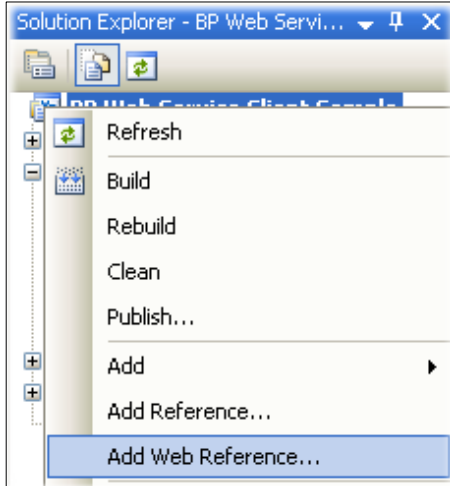


3. Select System.Web.Services and click OK

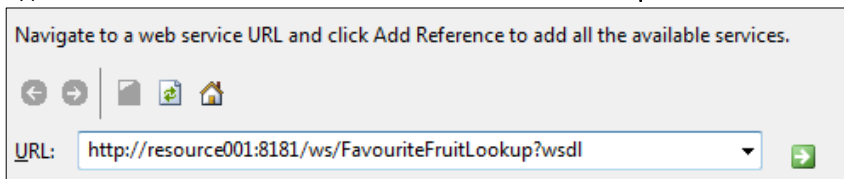


5.4.5. Add Web References to each of the WSDLs

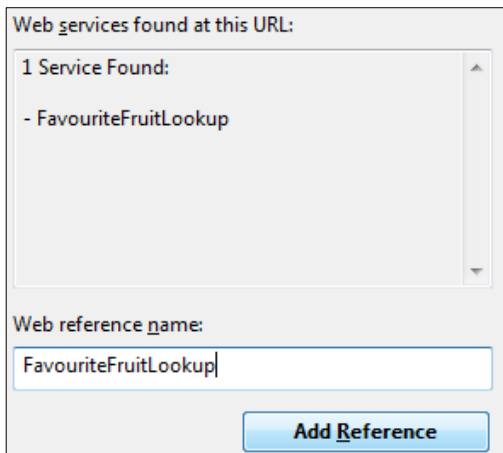
1. **Right-click** the project name within the **solution explorer** and select **Add Web Reference**.
(If using newer versions of Visual Studio select **Add Service Reference**, click **Advanced**, click **Add Web Reference**)



2. Type the **URL** of the WSDL for the **FavouriteFruitLookup**.



3. Visual Studio will then examine the WSDL document.
Set the name of the Web Reference to be **FavouriteFruitLookup**.



4. Repeat the above steps for the **NumericOperations WSDL** and name it **NumericOperations**.

5.4.6. Example Code: Interact with a Business Object based Web Service

Business Objects can comprise of many individual actions and each of these is made available via the exposed web service. For example the Business Object **NumericOperations** has two actions: **Add Numbers** and **Subtract Numbers**.

Business Object based web services must be initialized before use. There are two methods available:

- **Manual Initialisation:** Provides full control of the life cycle of the Business Object. Achieved by using the **Intialise** action to receive a Session ID which is then used to populate the **bplInstance** input parameter on subsequent actions.
- **Auto-Initialisation:** Reduces the number of steps required as part of calling a Business Object action as Blue Prism handles the initialization of the Business Object on behalf of the user. Achieved by setting the parameter **bplInstance** to have a value of **“auto”**.

When selecting which initialization mode to use the following should be considered:

- With auto-initialization, any global variables defined within the Blue Prism Business Object will remain initialized across each relevant web service request.
- If using manual initialization a session will be displayed within Control Room each time the Business Object is initialized where-as with auto-initialization, a single session will be used for all actions taken against a given web service.

Irrespective of the choice of initialization method, the steps required are as follows.

1. Create a sub procedure named **TestNumericWebService**

```
Private Sub TestNumericWebService()  
End Sub
```

2. Insert the following lines to display information in the console when the application runs and to create a new instance of the proxy class for interacting with the web service.

```
Console.WriteLine("Numeric Web Service Test")  
Dim NOS As New NumericOperations.NumericOperationsService
```

3. Add the following line which handles the authentication with Blue Prism

```
NOS.Credentials = New System.Net.NetworkCredential("admin", "admin")
```

The username and password must be valid for Blue Prism credentials for a user with permissions log in and run a Process from Control Room. The user will also require access to run a Business Object.

4. If using **manual initialization**, append the following line which carries out the initialization and returns the Session ID. This Session ID will be supplied when any of the Business Object action methods are requested to allow Blue Prism to distinguish between multiple concurrent running instances on the same Business Object.

```
Dim SessionID As String = NOS.Initialise()  
Console.WriteLine("Initialised")
```

If using **auto-initialization**, instead append the following line which, when used, will tell Blue Prism to handle the initialization automatically.

```
Dim SessionID As String = "auto" 'set string value as auto
```

5. Include the following code which calls the two different actions and outputs the results. Note that each pass either the Session ID returned by the initialize action, or the text “auto” if using auto-initialization

```
'Call the action method "Add Numbers"
Dim Result1 As Decimal = NOS.AddNumbers(SessionID, 3, 5)
Console.WriteLine("Added 3 and 5 to get " & Result1.ToString)

'Call the action method "Subtract Numbers"
Dim Result2 As Decimal = NOS.SubtractNumbers(SessionID, 3, 5)
Console.WriteLine("Subtracted 5 from 3 to get " & Result2.ToString)
```

6. Finally, if using **manual-initialization** it is necessary to carry out a clean-up task of closing the session. This allows Blue Prism to close the session and dispose of the associated resources.

```
Console.WriteLine("Cleaning up ...")
NOS.Cleanup(SessionID)
```

If using **auto-initialization** no clean-up actions are required.

The complete procedure should appear as follows:

- o **Manual** Initialisation

```
Private Sub TestNumericWebService()
    Console.WriteLine("Numeric Web Service Test")
    Dim NOS As New NumericOperations.NumericOperationsService

    NOS.Credentials = New System.Net.NetworkCredential("admin", "admin")
    Dim SessionID As String = NOS.Initialise()
    Console.WriteLine("Initialised")

    'Call the action method "Add Numbers"
    Dim Result1 As Decimal = NOS.AddNumbers(SessionID, 3, 5)
    Console.WriteLine("Added 3 and 5 to get " & Result1.ToString)

    'Call the action method "Subtract Numbers"
    Dim Result2 As Decimal = NOS.SubtractNumbers(SessionID, 3, 5)
    Console.WriteLine("Subtracted 5 from 3 to get " & Result2.ToString)

    Console.WriteLine("Cleaning up ...")
    NOS.Cleanup(SessionID)
End Sub
```

- o **Auto**-Initialisation

```
Private Sub TestNumericWebService()
    Console.WriteLine("Numeric Web Service Test")
    Dim NOS As New NumericOperations.NumericOperationsService

    NOS.Credentials = New System.Net.NetworkCredential("admin", "admin")
    Dim SessionID As String = "auto"
    'Console.WriteLine("Initialised")

    'Call the action method "Add Numbers"
    Dim Result1 As Decimal = NOS.AddNumbers(SessionID, 3, 5)
    Console.WriteLine("Added 3 and 5 to get " & Result1.ToString)

    'Call the action method "Subtract Numbers"
    Dim Result2 As Decimal = NOS.SubtractNumbers(SessionID, 3, 5)
    Console.WriteLine("Subtracted 5 from 3 to get " & Result2.ToString)

End Sub
```

5.4.7. Example Code: Interact with a Process based Web Service

Unlike Business Objects, Processes run once for each instance, only have one entry point and there are no explicit initialization or clean-up actions required.

1. Create a sub procedure named **TestFruitLookupWebService()**

```
Private Sub TestFruitLookupWebService()
End Sub
```

2. Use the following lines to display information in the console when the application runs and to create a new instance of the proxy class for interacting with the web service.

```
Console.WriteLine("FruitLookup Web Service Test")
Dim FFL As New FavouriteFruitLookup.FavouriteFruitLookupService
```

3. Add the following line which handles the authentication with Blue Prism

```
FFL.Credentials = New System.Net.NetworkCredential("admin", "admin")
```

The username and password must be valid for Blue Prism credentials for a user with permissions log in and run a Process from Control Room.

4. Add code which defines a string array of names and subsequently loops through each name and for each, queries the web service to find out what their favourite fruit is.

```
For Each Person As String In New String() _
{"Bashful", "Doc", "Dopey", "Grumpy", "Happy", "Sleepy", "Sneezy"}
    Dim Fruit As String = FFL.FavouriteFruitLookup(Person)
    Console.WriteLine(String.Format("{0}'s favourite fruit is {1}", Person, Fruit))
Next
```

The complete procedure should appear as follows:

```
Private Sub TestFruitLookupWebService()
    Console.WriteLine("FruitLookup Web Service Test")
    Dim FFL As New FavouriteFruitLookup.FavouriteFruitLookupService
    FFL.Credentials = New System.Net.NetworkCredential("admin", "admin")
    For Each Person As String In New String() _
        {"Bashful", "Doc", "Dopey", "Grumpy", "Happy", "Sleepy", "Sneezy"}
        Dim Fruit As String = FFL.FavouriteFruitLookup(Person)
        Console.WriteLine(String.Format("{0}'s favourite fruit is {1}", Person, Fruit))
    Next

    Console.WriteLine("End of FruitLookup Test")
End Sub
```

5.4.8. Example Code: Trigger the Interactions when the application is run

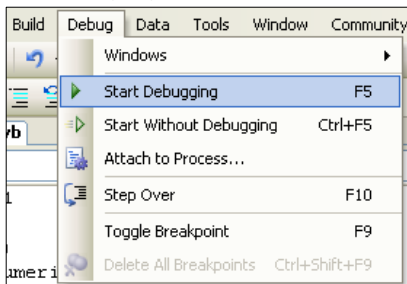
This example modifies the main procedure within the console application to trigger the above interactions with the Business Object based web service and the Process Based web service.

1. Modify the **main** procedure to call the sub procedures

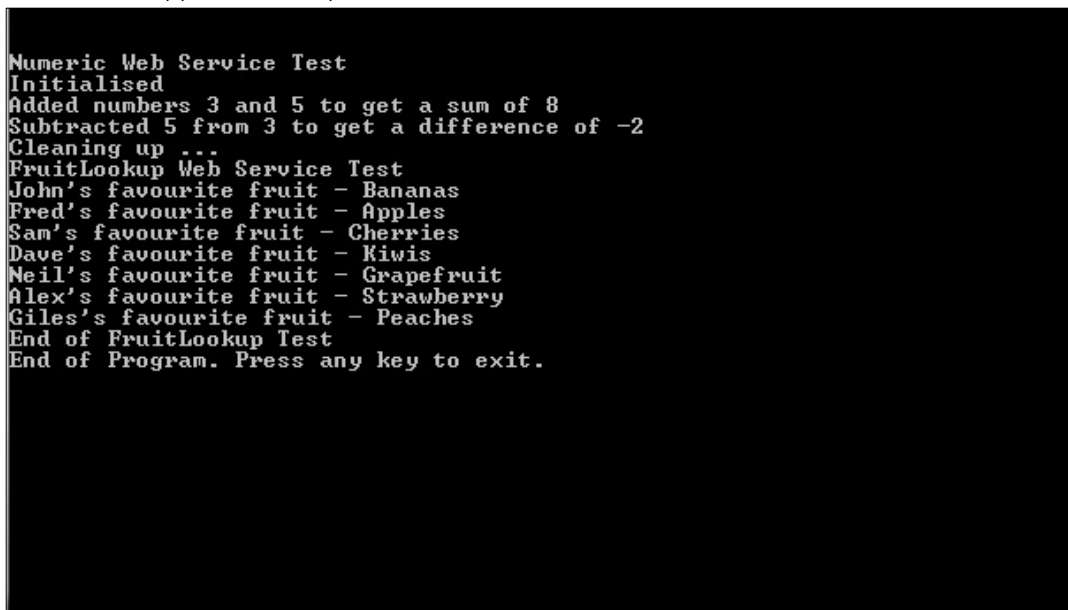
```
Sub Main()
    TestNumericWebService()
    TestFruitLookupWebService()

    Console.WriteLine("End of Program. Press any key to exit.")
    Console.ReadKey()
End Sub
```

2. Save the project in Visual Studio and run it using the debugging tool



3. The console application output should be similar to the screenshot below:



6. Frequently Asked Questions

1. What version of Blue Prism first contained wizards for exposing Business Objects and Processes as web services?

The wizard functionality for exposing Blue Prism Business Objects and Blue Prism Processes as web services was first introduced in version v4.2.35.

2. What is the timeout value for web services consumed by Blue Prism?

Prior to version 3.0.84, a third-party web service was configured to have a hard-coded timeout value of 100 seconds. Subsequent versions of Blue Prism allow this default to be configured for each web service.

3. Can Blue Prism handle binary web service data?

Introduced in version 3.0.76, Blue Prism supports binary data as inputs and outputs for both exposed and consumed web services. This provides the ability to pass files (documents, images, pictures etc.) as input or output parameters.

4. Can Blue Prism consume third-party web services presented over HTTPs?

Yes, however it is important that the Blue Prism runtime resources that will use such web services are configured to trust the certificate that is used to secure the web service. This may include ensuring a public certification authority issues the certificate or ensuring that the certificates are added to the appropriate zones on each runtime resource.

5. Can Blue Prism consume RESTful web services?

Support is primarily provided for a range of SOAP Web Services, however support for REST web services is provided for developers with appropriate programming experience through use of a series of example VBOs: REST, HTTP, JSON and Oauth.

6. Can Blue Prism handle structured web service data?

Introduced in version 3.5.33, Blue Prism supports nested collections. This allows Blue Prism to send and receive structured data when exposing or consuming web services.

7. What authentication is needed to access Blue Prism web services?

Blue Prism web services are secured using HTTP authentication. It is recommended that a Blue Prism user account is configured for each third-party provider that will be consuming exposed Blue Prism web services. The credentials for these users can then be used as part of the HTTP header authentication process.

8. What is the limit to the size of web service request that Blue Prism can make?

Blue Prism does not have a limit to the size of request that it can make. It should be noted however that web service servers typically impose a limit (commonly 4mb).

9. Is support provided for consuming WSDLs that include recursive definitions?

Recursive definitions within WSDLs are not supported by Blue Prism. An example snippet of a WSDL that includes a recursive definition for **DataCategory** is below:

```
<complexType name="DataCategory">
  <sequence>
    <element name="childCategories" type="tns:DataCategory" minOccurs="0"
maxOccurs="unbounded"/>
    <element name="label" type="xsd:string"/>
    <element name="name" type="xsd:string"/>
  </sequence>
</complexType>
```

7. Support

If further assistance is required whilst following this document please contact your Blue Prism Account Manager or Technical Support (support@blueprism.com).