ENTERPRISE ARCHITECT

User Guide Series

Integrate Data from External Providers

Author: Sparx Systems

Date: 2022-10-03

Version: 5.0

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

Integrate Data from External Providers

The Pro Cloud Server helps you to integrate the data from external providers into an Enterprise Architect model. Enterprise Architect is a team player and through its server-based integration capability helps you to create models that relate elements from a wide range of disparate tools, each of which might contain entities such as strategic objectives and Requirements through to implementation statements, work packages and configuration items. Enterprise Architect does not manage the master records for these items, but rather acts as an accumulator, bringing content into a single repository and allowing the items to be related. A variety of third-party providers can be integrated with Enterprise Architect, including:

- Application Lifecycle Management (formerly HP Quality Center)
- Jama Integration
- Jazz (interacts with:
  - IBM Rational DOORS Next Generation's requirements management tool
  - Rational Rhapsody Design Management (DM)
  - Rational Team Concert Change and Configuration Management (CCM)
  - Rational Quality Manager (QM))
- Jira and Confluence
- Polarion
- Azure DevOps / Team Foundation Server
- Wrike
- ServiceNow
- Autodesk
- Bugzilla
- Salesforce
- SharePoint
- Dropbox and
- Other Enterprise Architect models

See the Install and Configure Help topic for information on how to configure each provider. Walkthroughs are also available for Walkthrough: Jira Integration and Walkthrough: Polarion Integration integration.

When an item from an external provider is selected in the list the meta-data for the item will be displayed in the appropriate Enterprise Architect window. So any property-and-value type of information will be displayed in the Properties window, descriptions and comments will be displayed in the Notes window and discussions or posts will be displayed in the Discuss & Review window. For example, if Jira was the External Data source and the integration was listing User Stories, a Jira User Story property such as *Priority: Medium* would appear in the Properties window, the *Story description* would appear in the Notes window and the *comments* would appear in the Discuss & Review window.

Some meta-data such as collaboration information might not be available for all items and integrations, but where it is available the facility provides a uniform, cross-integration view of the meta-data, making it easy for Enterprise Architect users to understand the data from multiple providers and integrations without the need to leave the tool or grapple with vendor specific terminology. Enterprise Architect is performing the role of an accumulator, allowing information from a wide range of disparate sources to be related to the already rich set of architectural models in the tool, creating a view of how the information in these otherwise unrelated tools can be visualized. This removes the need for the bundle of static spreadsheets that organizations have traditionally used to relate pairs of items such as Test-Cases to Business Drivers, or Stakeholder Concerns to application services, and much more.

All integrations offer support for linking objects and elements from the external system into an Enterprise Architect client. The External Data window supports browsing the external provider's items and retrieving lists of elements and objects based on the provider's queries. Capabilities include:

- Link an Enterprise Architect element to an external object
- View external element properties
- View and in some cases add to, external object discussions
- Export links to WebEA URL's that correspond to the current model
- Open external items in a web browser
- Import elements
- Export elements

From Enterprise Architect Release 14.1 onwards it is possible to link a non-Cloud model to Integration Plug-ins configured on a Pro Cloud Server.

See the *Cloud Page* Help topic for configuration options.

Access

Ribbon	Specialize > Tools > System Integration > Open External Data To view a list of all elements that are linked to external items: Specialize > Tools > System Integration > Show All Linked Items
Context Menu	In a diagram or the Browser window, for elements that are already linked to an external item: Right-click on element > Specialize > External Properties
Keyboard Shortcuts	Alt+1 > System Integration

Features

Feature	Description
Pro Cloud Server Configuration	Each external provider must be configured on the Pro Cloud Server to enable connection. Multiple configurations can be made for each provider (such as connecting to two separate Jira servers).
Authorization	If the Integration Provider requires authorization you are prompted to enter your credentials. If the provider supports it, a new internet browser window will open

	and prompt you to log in to the Integration Provider and allow Enterprise Architect access to its resources. Alternatively, a simple dialog will pop up asking for your credentials, with the option to securely store them in the current model. If stored in the model the credentials will only be used for the current user.
Navigate External Provider	Find external items to link to by navigating the external provider.
External Item Details	Select an external item from the External Data window to see its properties, notes and discussions in the Properties window, Notes window and 'Discuss' tab of the Discuss & Review window.
Link External Items	External items can be linked to Enterprise Architect as a generic «ExternalReference» stereotype or as another element type.
Add WebEA Link	Hyperlinks to WebEA can be added to the external item so that you can quickly open the WebEA element from the external source. Right-click on a local linked element in the list and select 'Add WebEA Hyperlink to External Object'. This will update the external item with a link to the WebEA element that is linked to it in Enterprise Architect. Note that not all provider types have a 'link' mechanism. Where none exists, some providers might allow adding the link as a comment on the item. Ensure that the model has a valid WebEA address set in the model options.
Configuration	Each Integration Provider comes with a set of default mapping values which determines what type of local element is created in Enterprise Architect, as well as which fields are copied to the new element. These mappings are configurable via the 'Configure' option on the External Data window toolbar menu. You must have the 'Configure External Data Sources' permission to access this functionality.
Troubleshooting	• The System Output window will show any errors while attempting to retrieve data from the external providers • The Pro Cloud Server outputs log files for each external provider

Notes

- 'Integration' requires a Pro Cloud Server and is only available to Cloud models
- 'Integration' is currently provided with the Corporate, Unified and Ultimate Editions of Enterprise Architect
- If you select two or more elements at the same time, you can perform an operation on all of the selected elements at once
- For WebEA links to work a valid URL must be set for the WebEA address (see the *Cloud Page* Help topic)

Install and Configure

The Sparx Systems Pro Cloud Server product is installed using a standard Microsoft Windows Installer package that includes a number of optional components. One of these components is Integration Plug-ins, which is installed by default into the C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\ folder (assuming a 64 bit machine is being used).

Requirements

In order to install a particular Integration Plug-in you must have:

- A licensed Pro Cloud Server
- Physical network access to a server hosting the external data
- User credentials to access the external data
- Enterprise Architect Corporate, Unified or Ultimate Edition, v14 or later
  Nb: Enterprise Architect Trial edition provides read-only access to Integrations

What is in the Package?

The Integration Plug-ins folder initially consists of an Integration Server executable (SBPI.exe), a separate executable (*SBPI.exe) for each supported external product, a sample configuration file and a text file describing the manual installation and configuration steps.

Plug-In	Description
Integration (SBPI) Server	The SBPI.exe application acts as the interface between the Pro Cloud Server and each of the plug-ins, by translating Enterprise Architect requests, forwarding them to the appropriate plug-in and then returning the generated response to Enterprise Architect.
Application Lifecycle Manager Plug-in	The ALMSbpi.exe plug-in interacts with the MicroFocus Application Lifecycle Manager product (previously known as HP Quality Center).
Autodesk Plug-in	The AutodeskSbpi.exe Plug-in interacts with AutoCAD's file and management component AutoDesk.
Bugzilla Plug-in	The BugzillaSbpi.exe Plug-in interacts with the web based defect/bug tracking system that was originally developed and used by the Mozilla project and is now licensed under the Mozilla Public License agreement.
Confluence Plug-in	The ConfluenceSbpi.exe plug-in interacts with Atlassian's Team Collaboration Software. It is able to list *spaces* and link to *pages.* Note that the content of the Confluence HTML pages is not synchronized.
Dropbox Plug-in	The DropboxSbpi.exe plug-in interacts with Dropbox's web based file hosting service. It is able to list folders within Dropbox and link to individual files.
EA Plug-in	The EASbpi.exe Plug-in interacts with external Sparx Systems's Enterprise Architect Cloud-based repositories. It is able to browse the Package hierarchy or perform search based queries.

Jazz Plug-in	The JazzSbpi.exe Plug-in interacts with: • IBM Rational DOORS Next Generation's Requirement management tool • Rational Rhapsody Design Management (DM) • Rational Team Concert Change and Configuration Management (CCM) • Rational Quality Manager (QM)
Jira Plug-in	The JiraSbpi.exe plug-in interacts with Atlassian's issue tracking system. It is able to list a user's favorite filters (also known as *starred* filters). Each filter will then list all the Jira items returned by the filter.
Salesforce	The SalesforceSbpi.exe plug-in interacts with Salesforce's Customer Relationship Management system.
ServiceNow Plug-in	The ServiceNowSbpi.exe plug-in interacts with ServiceNow's asset management component of its Cloud-based enterprise management system.
SharePoint Plug-in	The SharePointSbpi.exe Plug-in interacts with Microsoft's web-based collaborative platform, SharePoint.
Azure DevOps / TFS Plug-in	The TFSSbpi.exe Plug-in interacts with Microsoft's Azure DevOps / Team Foundation Server (TFS) work items
Wrike Plug-in	The WrikeSbpi.exe Plug-in interacts with Wrike's project management system.

How to Set Up

The Integration framework consists of an Integration server (SBPI.EXE) application that starts one or more Plug-ins (such as DropboxSbpi.exe and JiraSbpi.exe). The Integration Server and each Integration Plug-in can be configured to run either on the same machine as the Pro Cloud Server or on completely different machines. In the simplest configuration the Integration server and all Integration Plug-ins are installed on a single server. There are two main advantages with this configuration:

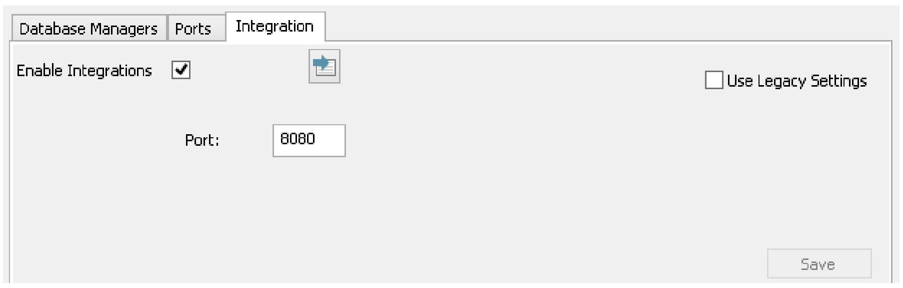
1. The Pro Cloud Server will automatically start (and stop) all configured Plug-ins whenever its Windows service is started (or stopped).
2. The Integration configuration GUI inbuilt into the Cloud Configuration client can be used to completely manage all aspects of the Integration configuration; see the *Steps - Simple* table.

However, if you elect to run the Integration Server or Integration Plug-ins on different machine(s) to the Pro Cloud Server, each of the individual Plug-ins must be manually configured as well as configured to start as Windows services with the correct parameters. See the *Steps - Manual* table.

Steps - Simple

The Integration configuration GUI included in the Cloud Configuration client removes most of the complexity involved in configuring the Integration Server and Data Provider(s), therefore this is the recommended method for most users. However, this configuration method is restricted to running all Integration components (*SBPI.exe) on the same server as the Pro Cloud Server.

Step	Description
Configuring Pro Cloud	Each installation of Sparx Systems Pro Cloud Server can be configured to

Server for Integration	communicate with a single Integration Server; the configuration options of this definition are defined as a series of registry settings, however the 'Integration' tab in the Pro Cloud Server Configuration Client allows the definition and maintenance of the Integration Server options without the need for you to manually manipulate the registry or configuration files. This image shows the definition of an Integration Server with the default settings:  Database Managers / Ports / Integration Enable Integrations [✓] Use Legacy Settings [ ] Port: 8080 Save Alternatively, the Integration Server and Plug-ins can be configured via the WebConfig interface. See the WebConfig - Integration Plug-ins topic for details.
Configuring Integration Data Providers	Each Data Provider (or Plug-in) is defined both as a series of registry entries in [HKEY_USERS\.DEFAULT\Software\Sparx Systems\SQLBridge\SBPI\Plugins\{unique} and as settings within a configuration file. Again, the 'Integration' tab in the Cloud Configuration Client allows the definition and maintenance of Integration Data Provider details without the need for you to manually manipulate the registry and configuration files.
Configuring the Firewall	In an effort to minimize Firewall rules needed to configure the Pro Cloud Server and its features, PCS version 4.1 now routes all requests for the Integration server (SBPI.exe) via the normal PCS ports, therefore if you are using PCS 4.1 or later there are no additional Firewall rules needed other than the ones for Enterprise Architect client to communicate to the Pro Cloud Server. For versions 3 and 4 of the PCS, the Integration server (SBPI.exe) typically must be granted access through any local firewall so that Enterprise Architect clients can connect to it. The Pro Cloud Server installer will automatically create a Firewall exception that allows any incoming requests to be passed through to SBPI.exe; however, the default settings should be reviewed and adjusted to suit your environment.

Steps - Manual

Important: these steps are only needed if the Integration components will run on different machine(s) to the Pro Cloud Server, otherwise the *Steps - Simple* table should be used.

Step	Description
Configuring Pro Cloud Server for Integration	Each installation of Sparx Systems Pro Cloud Server can be configured to communicate with a single Integration (or SBPI) server. These configuration options are defined as a series of registry settings. This is an example of all valid options for the Integration server: [HKEY_USERS\.DEFAULT\Software\Sparx Systems\SQLBridge\SBPI\Server] "Enabled"="true" "LocalPort"=dword:00001f90

	"UseLegacy"="false" "Arguments"="-port 8080 -protocol http" "Protocol"="https" "Server"="localhost" "Port"=dword:00001f90 "IgnoreSSLErrors"="true" "AttemptAutoDiscovery"="true" "ClientProtocol"="http" "ClientServer"="alternativeservername" "ClientPort"=dword:00001f90 • **Enabled** - true or false, representing the Port number that the SBPI server should be listening on, which value should match the value specified in the arguments; for example, dword:00001f90 (decimal 8080) • **LocalPort** - a hexadecimal value, representing the Port number that the Integration Server is listening on when the Use Legacy option is FALSE; for example, dword:00001f90 (decimal 8080) • **UseLegacy** - true or false, controls if the simple (false) or complex (true) set of configuration options should be used by the Integration Server • **Arguments** - not used in Pro Cloud Server 4.1 or later versions; in earlier versions this represents the arguments that are used to start the Integration server, which include the Port and Protocol the server should listen on - for example, "-port 8080 -protocol http" • **Protocol** - http or https, the protocol that should be used to communicate with the machine hosting the Integration server when the Use Legacy option is TRUE; this field is combined with the 'Server' and 'Port' to form the Integration Server's URL, which the Pro Cloud Server will send SBPI related requests to Note: The complete URL ({protocol}://{server-name}:{port} must be resolvable by the Pro Cloud Server machine • **Server** - the name (or IP number) of the machine hosting the Integration server when the Use Legacy option is TRUE (for example, yourdomain.com); this field is combined with the 'Protocol' and 'Port' to form the Integration Server's URL, which the Pro Cloud Server will send SBPI related requests to Note: The complete URL ({protocol}://{server-name}:{port} must be resolvable by the Pro Cloud Server machine • **Port** - a hexadecimal value, representing the Port number that the Integration Server is listening on when the Use Legacy option is TRUE - for example, dword:00001f90 (decimal 8080); this field is combined with the 'Protocol' and 'Server' to form the Integration Server's URL, which the Pro Cloud Server will send SBPI related requests to Note: The complete URL ({protocol}://{server-name}:{port} must be resolvable by the Pro Cloud Server machine • **IgnoreSSLErrors** - true or false, defines whether the SSL related errors that occur while communicating with the Integration Server component should be ignored when the Use Legacy option is TRUE • **AttemptAutoDiscovery** - true or false, defines if the Pro Cloud Server should automatically attempt to determine the Enterprise Architect client's network address and supply it to the Integration Server when the Use Legacy option is TRUE • **ClientProtocol** - http or https, defines the protocol that, when combined with the 'ClientServer' and 'ClientPort', forms the resolvable URL that Enterprise Architect clients can communicate to the Integration Server when the Use Legacy option is TRUE Note: The complete URL ({protocol}://{server-name}:{port} must be

	resolvable by the Enterprise Architect client machine • **ClientServer** - defines the server name (or IP number) that, when combined with the 'ClientProtocol' and 'ClientPort', forms the resolvable URL that Enterprise Architect clients can communicate to the Integration Server when the Use Legacy option is TRUE Note: The complete URL ({protocol}://{server-name}:{port} must be resolvable by the Enterprise Architect client machine • **ClientPort** - a hexadecimal value defining the Port number that, when combined with the 'ClientProtocol' and 'ClientServer', forms the resolvable URL that Enterprise Architect clients can communicate to the Integration Server when the Use Legacy option is TRUE; for example, dword:00001f90 (decimal 8080) Note: The complete URL ({protocol}://{server-name}:{port} must be resolvable by the Enterprise Architect client machine **Note:** From version 4.1 onwards of the Pro Cloud Server, the Integration Server does not need to have Firewall rules of its own so that Enterprise Architect clients can communicate with it. For PCS versions 3 and 4 the Integration Server (SBPI.exe) typically needs to be granted access through any local firewall so that clients can connect to it. The Pro Cloud Server installer will automatically create a Firewall exception that allows any incoming request to be passed through; however, the default settings should be reviewed and adjusted to suit your environment.
Configuring Data Providers	Each Data Provider is defined as a series of registry entries in: [HKEY_USERS\.DEFAULT\Software\Sparx Systems\SQLBridge\SBPI\Plugins\{unique} where {unique} is a unique UUID for the Data Provider. This is an example of a complete External Data Provider definition: [HKEY_USERS\.DEFAULT\Software\Sparx Systems\SQLBridge\SBPI\Plugins\{853489C1-4C22-4bad-9A8E-3098D07A3FC1}] "AutoStart"="true" "Enabled"="true" "Group"="" "Name"="Sparx Systems Sample account" "Port"=dword:00001f91 "Prefix"="jr1" "TypeKey"="jira" "Arguments"="-port 8081 -config jr.config" "Config"="jr1.config" • **AutoStart** - true or false, defines if the Integration Server (SBPI.exe) should maintain a running process (*sbpi.exe) for this Data Provider • **Enabled** - true or false, defines if the Integration Server (SBPI.exe) should allow communications to be forwarded to this Data Provider • **Group** - an optional value that can be used to 'sort' Providers into groups when displayed in Enterprise Architect • **Name** - a 'friendly' project name to describe the external Data Provider, which is displayed to all Enterprise Architect users; for example 'Sparx Systems Sample account' • **Port** - a hexadecimal value representing the Port number that the external data source expects to receive requests on, which value should match the value specified in the arguments; for example, dword:00001f91 (decimal 8081) • **Prefix** - a short unique name that is meaningful to the users and that prefixes

each link stored within the Enterprise Architect model; for example, jr1::10001 (where 10001 is the Jira object ID)

- **TypeKey** - defines the Provider type of the current Data Provider; only these supported values can be used: cint, csvc, alm, ad, bug, cflu, drop, ea, jazz, jira, sf, now, sp, tfs, wrike
- **Arguments** - deprecated from PCS 4.1 onwards, a dynamic arguments list is built from the individual settings; earlier versions used this field to define the arguments that should be used to start the Integration Plug-in, including the Port, Protocol and Config filename
- **Config** - deprecated from PCS 4.1 onwards, the config filename is now the prefix with a '.config'; in versions PCS 3 and PCS 4 this field was given an independent value that had to be unique

Each External Data Provider requires its own set of options to define the details of how the configured Plug-in connects to the External Data Source. These settings are stored in a .config file that resides in the same location as the Plug-in's .exe file. For example, using the above definition a 'jr1.config' would need to be created, and would contain information similar to this:

```
PROTOCOL=https
SERVER=example.com
PORT=443
BASEURL=myproject
USERNAME=
PASSWORD=
CREATEITEMS=false
MODIFYITEMS=false
POSTDISCUSSIONS=true
PROXY=10.0.0.0:3128
PROXYBYPASS=<local>;10.*
IGNORESSLERRORS=true
```

- **PROTOCOL** - the communication protocol, http or https
- **SERVER** - the name (or IP number) of the external data source's server, such as example.com
- **PORT** - the Port the external data source is configured to listen on, such as 443
- **BASEURL** - when the external data source supports multiple 'projects', the BaseURL property identifies which should be used; for example, for Enterprise Architect SBPI the BaseURL is the DB Alias as defined in the PCS
- **USERNAME** - optional - see Note1
- **PASSWORD** - optional - see Note1
- **CREATEITEMS** - defines if Enterprise Architect users can create items in the External Data Source - see Note2; default value: false
- **MODIFYITEMS** - defines if Enterprise Architect users can modify existing items in the External Data Source - see Note2; default value: false
- **POSTDISCUSSIONS** - defines if Enterprise Architect users can add Discussions to items within the External Data Source - see Note2; default value: true
- **PROXY** - optional - the server name or IP number and Port of the proxy server, such as 10.0.0.0:3128
- **PROXYBYPASS** - optional - a semi colon separated list of IP numbers that should not be sent through the proxy; for example, <local>;10.*
- **IGNORESSLERRORS** - optional - a boolean value to ignore any SSL

	certificate errors **Note1** • If the external server supports OAuth 2 authentication (Autodesk, Dropbox and Wrike), when accessing from within Enterprise Architect, a browser window will open and prompt you to log in to the external account and allow Enterprise Architect to access your account • Enterprise Architect never sees your credentials in this process; instead it is provided with a unique token to access the external account • If the external server allows basic authentication, then the username and password can be optionally set in the configuration file • If the username and password are not specified in the configuration file, Enterprise Architect will prompt you for credentials to access the external data **Note2** • Not all external products support the creation and modification of their objects; for example, Dropbox does not
Configuring Models for External Data Providers	In order for users of an Enterprise Architect model to connect to a given External Data Provider, a 'binding' between the Enterprise Architect model and the External Data Provider must be configured. This is performed by a series of registry settings in [HKEY_USERS\.DEFAULT\Software\Sparx Systems\SQLBridge\SBPI\Bindings\{unique}] for each model/External Data Provider combination. In this string, {unique} is an 8-digit hexadecimal number uniquely identifying the binding. For example: [HKEY_USERS\.DEFAULT\Software\Sparx Systems\SQLBridge\SBPI\Bindings\B6EE6851] "LocalModel"="eaexample" "Plugin"="{853489C1-4C22-4bad-9A8E-3098D07A3FC1}" • LocalModel - the DB Alias of the model, such as 'eaexample' • Plugin - the UUID of the external data source; for example, {853489C1-4C22-4bad-9A8E-3098D07A3FC1}

Special Notes

Using SSL at the Integration Server or Provider Level

If you choose to implement the HTTPS protocol at either the Integration Server level or Integration Provider level, the Integration Executables (*SBPI.EXE) will require a 'server.pem' file in the same folder as themselves; therefore, if using the default installation path this would mean 'server.pem' should be placed into the C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\ folder.

See the section *SSL Certificates* in the Add a Port Definition Help topic and the Self-Signed SSL Certificates Help topic for more information on how to create a valid 'server.pem' file.

The Integration Plug-in for Enterprise Architect Requirements:

- The BaseURL as the model's DB Alias
- The defined model configured as 'Enabled' and 'Enable Pro Features (OSLC, WebEA and Integration)' in the Pro Cloud Server

Integration Plug-ins

The Integration Plug-in components represent a set of Plug-ins (or extensions) for the Pro Cloud Server that enable data from external providers to be displayed within Enterprise Architect. These components are also known as the Server Based Plug-in Interface (SBPI).

The Integration Plug-in components of the Pro Cloud Server run as a series of executables (.exe files) that do not have a graphical user interface (GUI) themselves; however, the topics of this section describe the configuration editor screen(s) that are included in the Pro Cloud Server's Configuration Client that can be used to configure the Integration Plug-ins.

The main screen of the Configuration Client (installed as part of the Pro Cloud Server) includes an 'Integration' tab that is divided into two main sections. The top half defines the details of the Integration Server (or SBPI Server) with the lower half displaying the details of the various 'Data Providers'; each of these sections is described in greater detail in this topic.

Note 1: The Integration feature is only available in licensed editions of the Pro Cloud Server: Team, Enterprise or Token.

Note 2: If you are using the Token edition of the Pro Cloud Server, before configuring Integrations ensure you have enabled the required Integration Provider/s via the Token Allocation options; see the Manage Allocations Help topic.

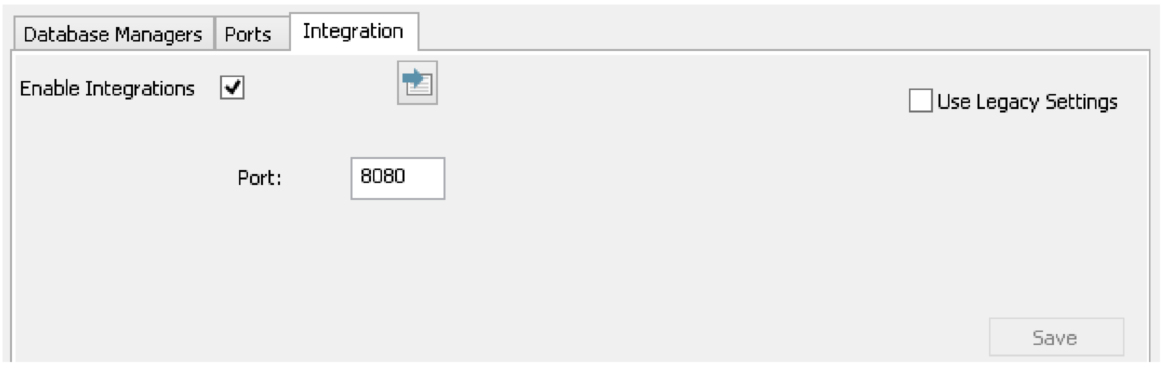
Note 3: If you are using the Enterprise Architect Trial edition, the Integration Plug-ins are read-only and will not update any stored data

Note 4: Any changes to the Integration Server or Plug-ins (made either through the Configuration Client or manually) will require a restart of the Pro Cloud Server

Alternatively, the Integration Server and Plug-ins can be configured via the WebConfig interface. See the WebConfig - Integration Plug-ins Help topic for details.

Detailed Walkthroughs are also available for Walkthrough: Jira Integration and Walkthrough: Polarion Integration integration specifically.

Integration Server



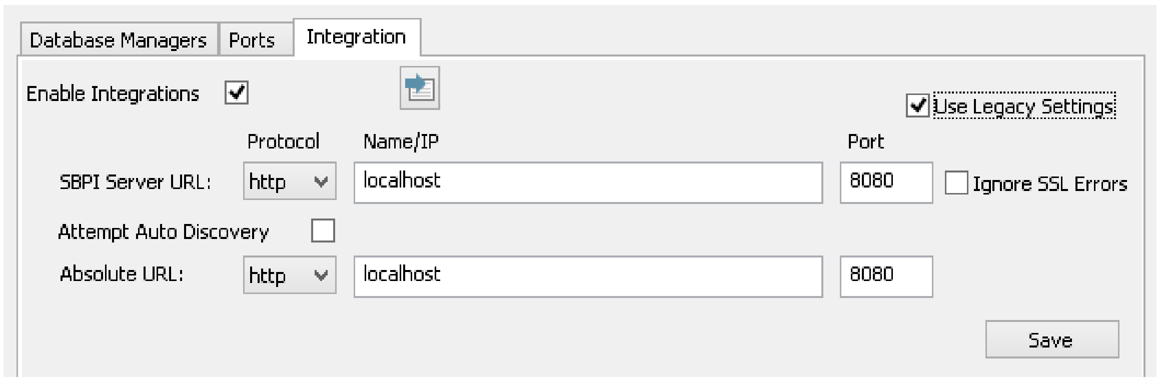
Enable Integrations ✓

Use Legacy Settings ☐

Port: 8080

All requests for external data from Enterprise Architect are sent to the Integration Server (via the Pro Cloud Server), which decides which Data Provider (or SBPI Plug-in) will action the request.

To make the Integration Server configuration as easy as possible for the majority of users, version 4.1 of the Pro Cloud Server has introduced a simplified set of options; however, if the default configuration does not suit your environment, select the 'Use Legacy Settings' option to show all the available options.



Enable Integrations ☑

☑ Use Legacy Settings

	Protocol	Name/IP	Port	
SBPI Server URL:	http	localhost	8080	☐ Ignore SSL Errors
Attempt Auto Discovery ☐				
Absolute URL:	http	localhost	8080	

Note: The Integration Server settings are only mandatory when the 'Enable Integrations' flag is checked. The combination of Protocol, Name and Port must match the machine hosting the Integration Server (in this case the Pro Cloud Server) in order for Enterprise Architect clients to be able to communicate with it.

Item	Description
Enable Integrations	Default value: Unchecked This checkbox enables (or disables) Integrations for the current Pro Cloud Server installation. When this value is selected it implies that the other fields ('Protocol', 'Name', 'Port' and 'Path') are now mandatory and the Integration Server definition can not be saved without a value in each field.
Load Defaults	This button loads the default values for the Integration Server settings into any empty field. **Note:** If a field already has a value, then the Load Defaults button will not change or update this existing value
Use Legacy Settings	Default value: Unchecked This checkbox determines if the simplified or complex set of options should be shown. The name refers to the fact that the earlier versions of the Pro Cloud Server required users to fill in the complex set of options. In later versions of the Pro Cloud Server the set of options needed was greatly reduced by making some assumptions that will be true for the majority of environments. When this value is selected it implies that the other fields (SBPI Server URL: 'Protocol', 'Name/IP' and 'Port') are now mandatory and the Integration Server definition can not be saved without a value in each field.
Port	Default value: 8080 This option is only shown when the 'Use Legacy Settings' option is not selected. It defines the Port that the Integration Server is listening to for requests from the Pro Cloud Server. Ensure that there is no other application or service already using the chosen Port.
SBPI Server URL: Protocol	Default value: HTTP This option is only shown when the 'Use Legacy Settings' option is checked. This field defines the SBPI Server's Protocol, combined with the 'SBPI Server Name/IP' and 'SBPI Server Port'; these settings form the Integration Server's URL that the Pro Cloud Server will send SBPI related requests to. Note: The complete URL ({protocol}://{server-name}:{port} must be resolvable by the Pro Cloud Server machine.

SBPI Server URL: Name/IP	Default value: localhost This option is only shown when the 'Use Legacy Settings' option is checked. This field defines the SBPI Server's machine name or IP, combined with the 'SBPI Server Protocol' and 'SBPI Server Port' these settings form the Integration Server's URL that the Pro Cloud Server will send SBPI related requests to. Note: The complete URL ({protocol}://{server-name}:{port} needs to be resolvable by the Pro Cloud Server machine.
SBPI Server URL: Port	Default value: 8080 This option is only shown when the 'Use Legacy Settings' option is checked. This field defines the SBPI Server's Port, combined with the 'SBPI Server Protocol' and 'SBPI Server Port'; these settings form the Integration Server's URL that the Pro Cloud Server will send SBPI related requests to. Note: The complete URL ({protocol}://{server-name}:{port} needs to be resolvable by the Pro Cloud Server machine.
Ignore SSL Errors	Default value: unchecked This option is only shown when the 'Use Legacy Settings' option is checked. This value defines if SSL related errors that occur while communicating with the Integration Server component should be ignored. The most common SSL errors are related to self-signed certificates; by default a client does not treat self-signed certificates as being secure, therefore in order to connect to servers that use self-signed certificates this option should be checked.
Attempt Auto Discovery	Default value: checked This option is only shown when the 'Use Legacy Settings' option is checked. This option defines if the Pro Cloud Server should automatically attempt to determine the Enterprise Architect client's network address and supply that to the Integration Server. This option was introduced in Pro Cloud Server 4.1.40; earlier versions were not capable of discovering the client's network address. When 'Attempt Auto Discovery' is checked, the setting is the 'Fallback URL', which will be used only if the Auto Discovery fails. When 'Attempt Auto Discovery' is unchecked, the setting is the 'Absolute URL' which will be used in all cases.
Absolute URL: Protocol	Default value: HTTP This option is only shown when the 'Use Legacy Settings' option is checked. This field defines the Protocol that when used in combination with the Absolute Server Name and Port form the resolvable URL that Enterprise Architect clients can communicate to the Integration server. Note: The complete URL {protocol}://{server-name}:{port} needs to be resolvable by Enterprise Architect client machines. In some environments server URLs are redirected to completely different locations, therefore to handle this situation the absolute URL is returned to Enterprise Architect clients so that they can communicate to the Integration Server.
Absolute URL: Name/IP	Default value: {empty} This option is only shown when the 'Use Legacy Settings' option is checked. This field defines the Server name/IP that when used in combination with the Absolute Protocol and Port form the resolvable URL that Enterprise Architect clients can communicate to the SBPI server. Note: The complete URL {protocol}://{server-name}:{port} needs to be resolvable

	by Enterprise Architect client machines. In some environments server URLs are redirected to completely different locations, therefore to handle this situation the absolute URL is returned to Enterprise Architect clients so that they can communicate to the SBPI Server
Absolute URL: Port	Default value: 8080 This option is only shown when the 'Use Legacy Settings' option is checked. This field defines the Port that when used in combination with the Absolute Protocol and Server Name form the resolvable URL that Enterprise Architect clients can communicate to the SBPI server. Note: The complete URL {protocol}://{server-name}:{port} needs to be resolvable by Enterprise Architect client machines. In some environments server URLs are redirected to completely different locations, therefore to handle this situation the absolute URL is returned to Enterprise Architect clients so that they can communicate to the SBPI Server
Save	This button saves any pending changes to the Integration Server settings; it is only enabled when there are unsaved changes.

Data Providers

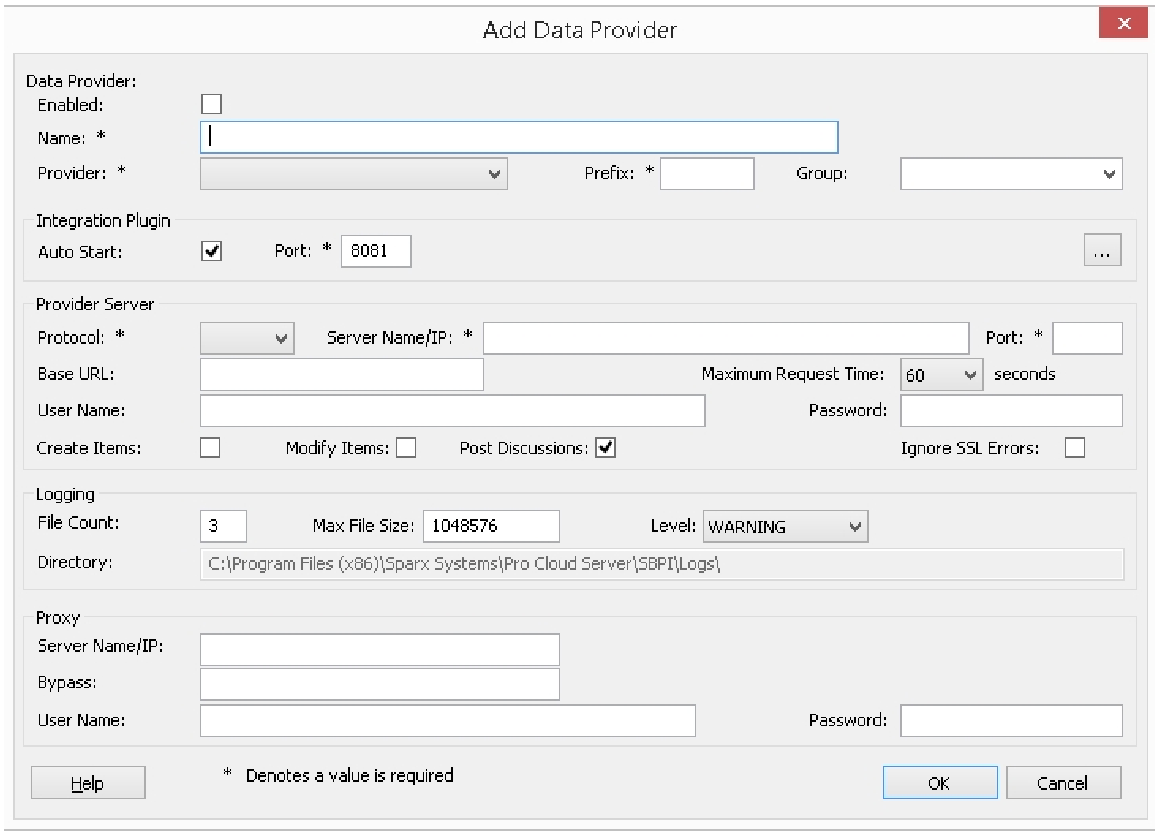
A single Pro Cloud Server can support any number of external Data Providers, and in turn each of the external Data Providers can be available to as many models as are supported by the installation's license. Note that only Pro-enabled repositories (or database managers) can access external data providers. A Pro-enabled database manager is one that has the 'Enable Pro Features (OSLC, WebEA and Integration)' option checked.

Option	Description
Defined Providers	This controls lists a summary of all defined external Data Providers. Double-clicking an existing item will display the Edit Data Provider screen.
Add...	This button displays the Add Data Provider screen to allow the entry of a new external Data Provider.
Edit...	This button displays the Edit Data Provider screen to allow the modification an existing external Data Provider definition. **Note:** This button is only enabled when there is a Data Provider selected.
Remove	This button permanently deletes the selected Data Provider and all of its Bindings. **Note:** This button is only enabled when there is a Data Provider selected.
<'plug-in name'> is bound to:	This control displays a check list of all Database Managers that are Pro-enabled (that is, the 'Enable Pro Features (OSLC, WebEA and Integration)' option is selected). A selected Database Manager allocates the current Data Provider to the given repository.
Check All	This button is a quick and easy way to allocate all defined Pro-enabled Database Managers to the currently selected Data Provider.
Uncheck All	This button is a quick and easy way to de-allocate all defined Pro-enabled Database Managers from the currently selected Data Provider.

Add/Edit Data Provider

The Add/Edit Data Provider screen will be shown whenever you request to create a new Data Provider or edit an existing one. The screen's behavior will be identical in either mode, the one difference being that the details of the selected Data Provider will be displayed when the screen loads in 'Edit' mode.

When you choose to add a new Data Provider, the screen will initially display as shown:



Add Data Provider

Data Provider:
Enabled: ☐
Name: *
Provider: * Prefix: * Group:

Integration Plugin
Auto Start: ✓ Port: * 8081

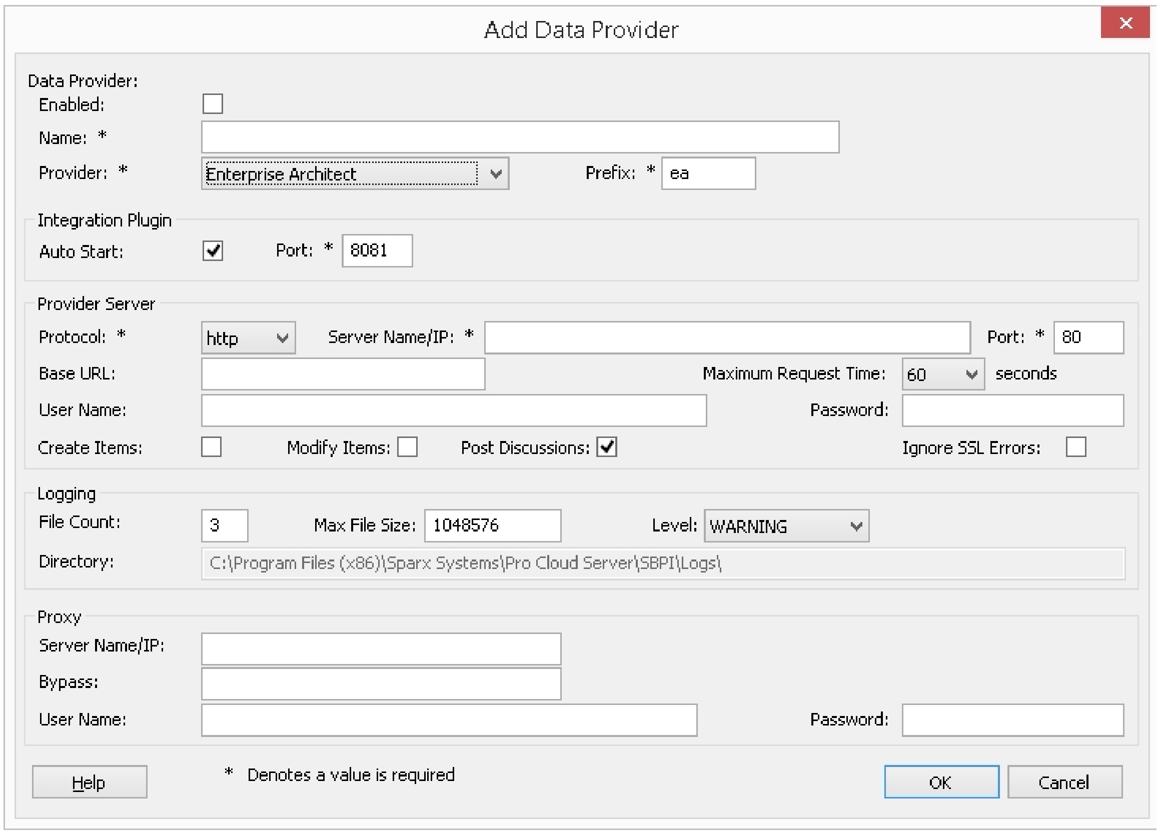
Provider Server
Protocol: * Server Name/IP: * Port: *
Base URL: Maximum Request Time: 60 seconds
User Name: Password:
Create Items: ☐ Modify Items: ☐ Post Discussions: ✓ Ignore SSL Errors: ☐

Logging
File Count: 3 Max File Size: 1048576 Level: WARNING
Directory: C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\Logs\

Proxy
Server Name/IP:
Bypass:
User Name: Password:

Help * Denotes a value is required OK Cancel

Wherever possible the Add/Edit Data Provider screen will fill each field with a default value in order to make it easier for you; if the default values are not correct, simply overwrite them. To this end, whenever the 'Provider' value is changed, a number of other fields will be set to the default values for the new Provider, such as the Data Provider's Prefix, as shown here when Enterprise Architect is selected:



Data Provider:

Enabled:	☐		
Name: *			
Provider: *	Enterprise Architect	Prefix: *	ea

Integration Plugin

Auto Start:	✓	Port: *	8081

Provider Server

Protocol: *	http	Server Name/IP: *		Port: *	80
Base URL:		Maximum Request Time:	60 seconds		
User Name:		Password:			
Create Items: ☐	Modify Items: ☐	Post Discussions: ✓	Ignore SSL Errors: ☐		

Logging

File Count:	3	Max File Size:	1048576	Level:	WARNING
Directory:	C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\Logs\				

Proxy

Server Name/IP:			
Bypass:			
User Name:		Password:	

* Denotes a value is required

Note: When defining a Data Provider there are a number of mandatory fields (denoted with a '*') that require a value before the Data Provider can be saved.

Data Provider settings

Item	Description
Enabled	Default Value: unchecked This value controls if the Integration Server (SBPI.exe) should allow communications to be forwarded to this Data Provider.
Name	This value represents the friendly name for the current Data Provider. It will be shown to users of Enterprise Architect.
Provider	This value specifies the Provider type of the current Data Provider; only supported values can be used. As mentioned, whenever this value is changed a number of other fields will be automatically updated to make data entry easier for the user.
Prefix	This value represents a short unique name to identify the current Data Provider. This value will be saved against every Enterprise Architect element that is linked to an external element.
Group	This field is available only when the 'Custom Integration' provider type is selected. Typically in Enterprise Architect, integrations are grouped by the Provider type (such as 'Jira' or 'Dropbox'). For Custom Integrations, the Grouping is applied based on the value entered in this field.
Max Request Time	This field is available only when the 'Custom Service' provider type is selected.

	This value represents the maximum amount of time (in seconds) that calls to the Provider system will wait before timing out. Increasing this value is particularly useful when the Provider systems are remote or have slow performance. The default value is 60 seconds. Note: This is a new setting introduced in Pro Cloud Server v4.2 build 65.

Integration Plug-in settings

This group of settings defines the relationship between the Integration Plug-in and the Integration Server and how the two interact.

Item	Description
Auto Start	Default Value: Checked This value controls if the Integration Server (SBPI.exe) should maintain a running process (*sbpi.exe) for this Data Provider. For standard Integration Plug-ins we would expect this value to be checked; however, for custom ones it might be useful to not auto start the Plug-in and instead manually control when it is running, particularly while developing it.
Port	Default Value: 8081 This value represents the Port that the Integration Server should use to communicate with the Integration Plug-in. The Integration Server assumes the Port is relative to the current machine (or http://localhost:{port}). Each Integration Plug-in requires a unique Port number, which will be checked when an attempt is made to save the Plug-in's settings. However, the Pro Cloud Server can only check what Ports are in use by the Pro Cloud Server itself and its components, there is still a possibility of a Port clash with other applications.
Protocol	This value represents the protocol that the Integration Server should use to communicate with the Integration Plug-in; in the simplest configuration this will be 'HTTP'.
DLL Path	This field is only available for 'Custom Service', 'Custom Integration' and 'Translator' provider types. This value represents the physical path to the custom DLL, that will be called by the CustomService.exe or CustomIntegration.exe Data Provider processes. As a convenience, the application populates the drop list with all .DLL files that are present in the .\SBPI\Custom\ folder (C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\Custom\).

Custom Properties

The Custom Properties section is only shown for Data Providers of type 'Custom Service'.

Item	Description
List of Custom Properties	Lists the current properties of the Custom Data Provider.

Add	This button shows the 'Add SBPI Provider Custom Property' screen and allows entry of a new custom property, defined as an Item/Value pair.
Edit	This button shows the 'Edit SBPI Provider Custom Property' screen and allows modification of an existing custom property definition.
Delete	This button permanently removes the selected Custom Property from the current Data Provider.

Provider Server settings

This group of settings defines the details of the external data provider, such as Dropbox, Jira or TFS. The Integration Plug-in will use the defined details to connect to the remote system and retrieve data from it so that it can be sent back to Enterprise Architect. The combination of 'Protocol', 'Server Name 'and 'Port' must be resolvable by the server hosting the Integration Plug-in, which in the simplest case is the Pro Cloud Server.

Item	Description
Protocol	This value represents the protocol that the Integration Plug-in should use to communicate with the external Data Provider. Note: Most online Cloud-hosted providers require https and Port 443.
Server Name/IP	This value represents the Server Name or IP address that the Integration Plug-in should use to communicate with the external Data Provider.
Port	This value represents the Port that the Integration Plug-in should use to communicate with the external Data Provider. Note: Most online hosted accounts operate via https, which requires a Port of 443. Self-hosted servers will depend on the individual server configuration. • Jazz: server defaults to using http Port 9080 or https 9443 • Jira: self-hosted server default Port is http Port 8080 • Confluence: self-hosted server default Port is http Port 8090
Base URL	Some external Data Providers allow for the concept of multiple repositories at a single location; for such Data Providers this field acts as a filter. For example, it is possible to use another Enterprise Architect model as an external data source; in this case the 'Protocol', 'Server' and 'Port' determine the Pro Cloud Server (for example, http://myEAServer:804) while the 'Base URL' is the DB Alias of the repository (myModel). Thus: http://myEAServer:804/myModel Details for specific providers: • Confluence: online Cloud hosted accounts require this to be 'wiki'; self-hosted servers will be dependent on the individual server set-up • Jira: for online Cloud hosted accounts this should be blank; self-hosted servers will be dependent on the individual server set-up • Dropbox: this should be left blank • Enterprise Architect: this should be the model alias to be connected to
Maximum Request Time	This value represents the maximum amount of time (in seconds) that calls to the

	Provider system will wait before timing out. Increasing this value is particularly useful when the Provider systems are remote or have slow performance. The default value is 60 seconds. Note: This is a new setting introduced in Pro Cloud Server v4.2 build 65.
User Name	This value represents the user name that should be used (in combination with the password) to access the data within the external system. If a value is defined in this field all Enterprise Architect users of the current Data Provider will use the same set of credentials to read the external system. Otherwise, if the user name field is left empty, Enterprise Architect will prompt each user independently for their credentials. Note: For many online Cloud accounts, the username can be an email address for the account.
Password	This value represents the matching password for the user name specified in the 'User Name' field. **Note:** For Atlassian Cloud hosted accounts (Jira, Confluence) the use of regular passwords has been replaced with an API Key. See the Atlassian documentation on how to obtain an API key and paste the API key into the password field.
Create Items	Default value: unchecked This checkbox controls if Enterprise Architect users are able to create new items within the external system.
Modify Items	Default value: unchecked This checkbox controls if Enterprise Architect users are able to modify the details of external items within the external system.
Post Discussions	Default value: checked This checkbox controls if Enterprise Architect users are able to create discussions against elements within the external system.
Ignore SSL Errors	Default value: unchecked This checkbox controls if the Integration Plug-in should ignore SSL related errors that occur as a result of communicating with the external system.

Logging settings

This group of settings define how the Integration Plug-in will create entries in its log file(s). Since the Integration Plug-ins run without user intervention, it is important for them to be able to write any messages or reports of potential problems to a log file. However, special consideration must be given to ensuring that performance isn't impacted by writing to such files, which is possible if the log file continually grows without limits. It is also unreasonable to expect that manual intervention should be required to ensure that the size and number of log files don't impact performance. For these reasons each Integration Plug-in has its own set of log files and configuration options to manage those log files.

The current log file will always include a '1' on the end of its filename. A new log file will be created whenever the Integration Plug-in is started or the physical size of the current log file reaches the Max File Size. When this occurs all existing files will be 'rolled over', meaning their file numbers will be incremented and a new '{filename}1.log' created. If the roll over process causes more files to exist than the 'File Count' setting allows, these additional files will be deleted.

Item	Description

File Count	Default value: 3 This value represents the 'rolling' number of log files that should be retained for the current Integration Plug-in.
Max File Size	Default value: 1048576 This value represents the maximum size, in number of bytes, that a log file can reach before a new log file is created.
Level	Default value: WARNING This value represents the maximum level of messages that should be written to the log file. The levels are: OFF, FATAL, WARNING, INFO and SYSTEM. A defined log level is inclusive of all lower levels; therefore if a Plug-in is set to a log level of INFO then all FATAL, WARNING and INFO messages will be written to its log file.
Directory	(READ-ONLY) This field displays the physically location of the directory into which the log files for the current Data Provider will be saved.

Proxy settings

This group of settings defines how the Integration Plug-in should communicate with the External Data Provider when they are separated by a Proxy Server.

Item	Description
Server Name/IP	The server name (or IP) and Port number of the Proxy Server; for example, proxyserver.com:3131
Bypass	This field provides a mechanism for certain addresses to bypass the proxy; it accepts multiple values separated by semi-colons (;). For example, 192.168.*;*.localdomain.com;www.mydomain.com
User Name	If the Proxy Server requires credentials, this field provides the user name.
Password	If the Proxy Server requires credentials, this field provides the user password.

Walkthrough: ServiceNow Integration

This walkthrough helps you to set up and use the integration capability of Pro Cloud Server to integrate your Enterprise Architect models with the corresponding ServiceNow tables.

Using the integration with ServiceNow you can:

- Link Enterprise Architect elements with their counterparts in ServiceNow
- Create Enterprise Architect elements based on ServiceNow items, and vice-versa
- Perform a traceability analysis

Prerequisites

In order to set up integration between ServiceNow and Enterprise Architect, you must first:

- Be using a licensed installation of Pro Cloud Server with the Integration Plug-ins component
- Select the 'Enable Pro Features (OSLC, WebEA and Integration)' option in the Pro Cloud Server configuration for each Database Management System hosting your Enterprise Architect models
- Have network access between the PCS server and ServiceNow
- Have user credentials to access the external data
- Be using Enterprise Architect Corporate, Unified or Ultimate edition v14 or later

We will assume that the Integration Server and the ServiceNow Integration Plug-in are to run on the same machine as the Pro Cloud Server.

Minimum required permissions

Read access is required on these tables:

- **sys_filter** - lists the custom filters that determine what is displayed in Enterprise Architect
- **sys_db_object** - list of tables to group the filters by the main base table used
- **sys_dictionary** - for the display name of table fields
- Any table that the custom filter uses

To enable 'push' synchronization, the user will require write access to the tables as well.

Read access can be enabled by adding an Access Control (ACL) rule on the table.

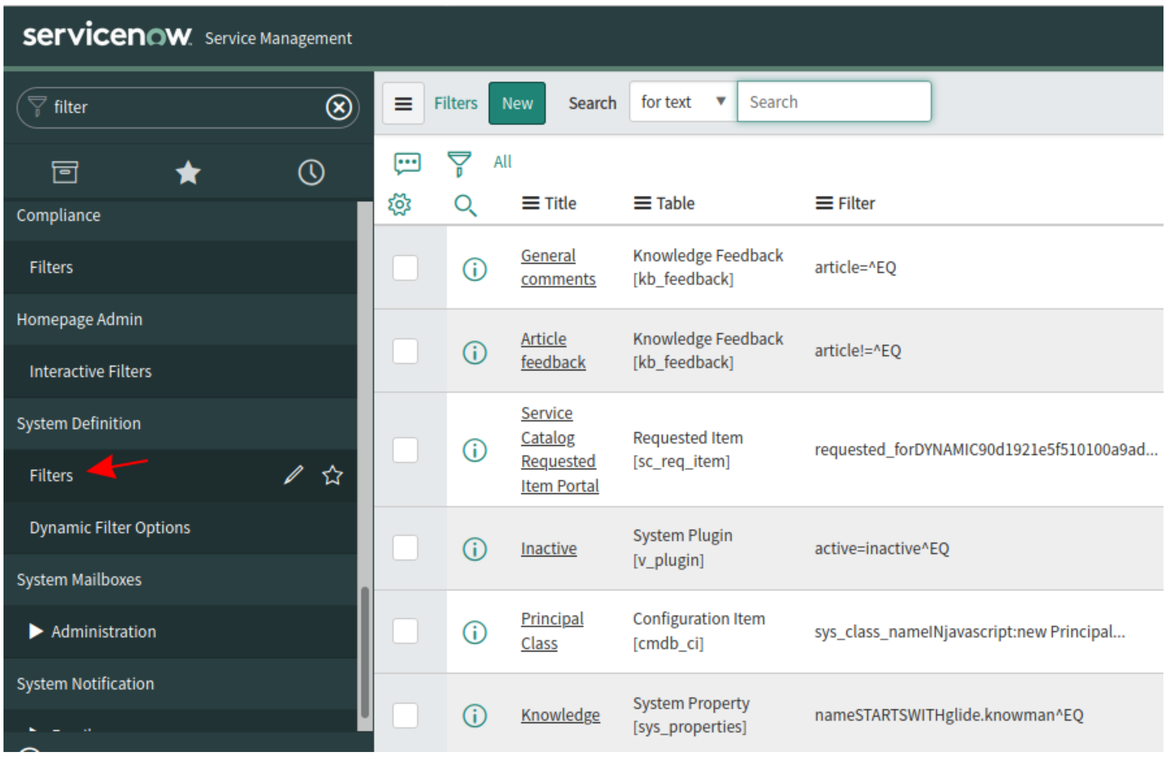
A second rule will be required to enable read access on all fields in the table (e.g. **table.***).

The Access Control rules should have a required Role assigned, and then that role can be given to a group or individual user.

Note: On some tables it is required to 'Elevate role' from a System Administrator to a 'Security Administrator - security_admin' to be able to edit Access Controls.

Define ServiceNow table filters

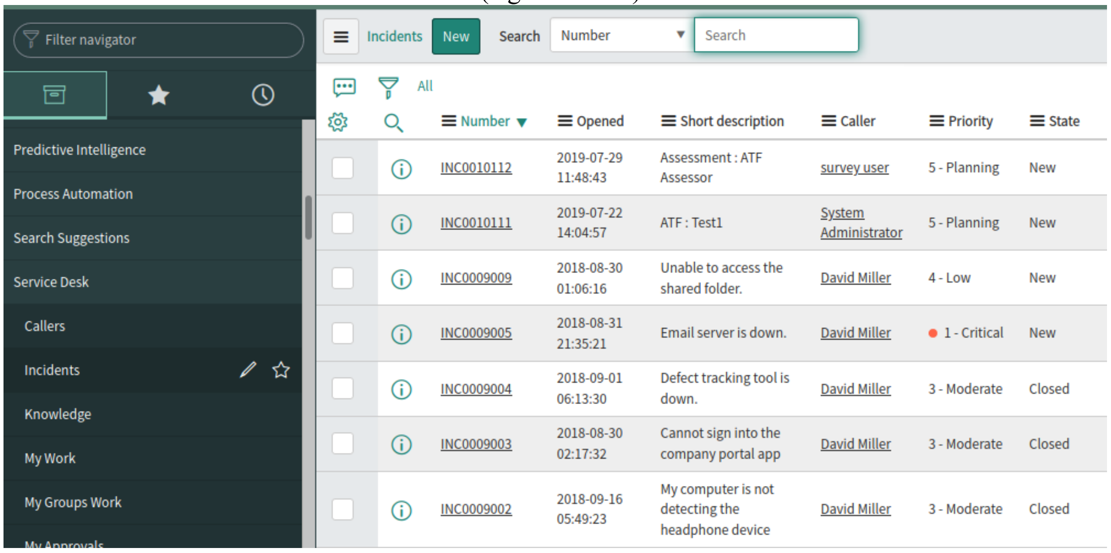
The ServiceNow integration allows access to items via saved table filters. By default ServiceNow typically includes a number of predefined global filters. You can browse all your defined filters in ServiceNow using the 'System Definition | Filters' option.



Title	Table	Filter
General comments	Knowledge Feedback [kb_feedback]	article=^EQ
Article feedback	Knowledge Feedback [kb_feedback]	article!=^EQ
Service Catalog Requested Item Portal	Requested Item [sc_req_item]	requested_forDYNAMIC90d1921e5f510100a9ad...
Inactive	System Plugin [v_plugin]	active=inactive^EQ
Principal Class	Configuration Item [cmdb_ci]	sys_class_nameINjavascript:new Principal...
Knowledge	System Property [sys_properties]	nameSTARTSWITHglide.knowman^EQ

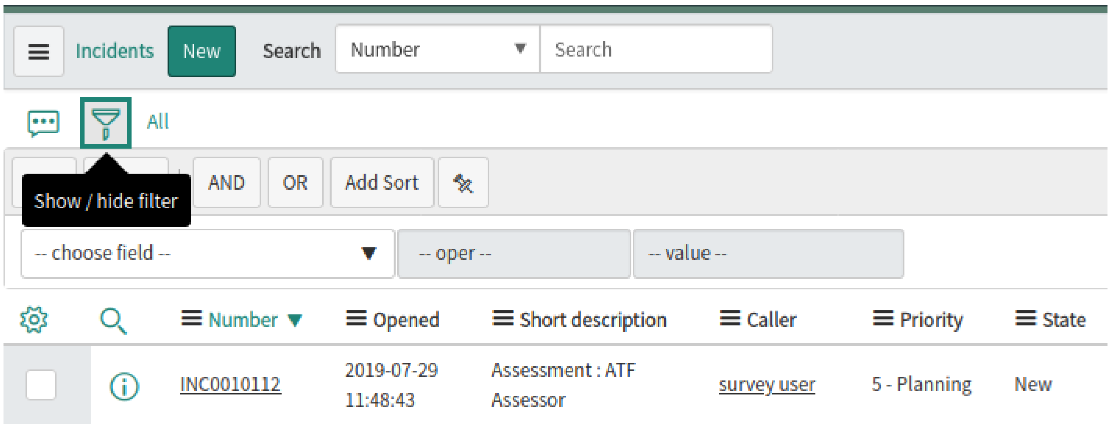
You can create new filters in ServiceNow by following these steps:

1. Locate the relevant data within ServiceNow (e.g. Incidents)



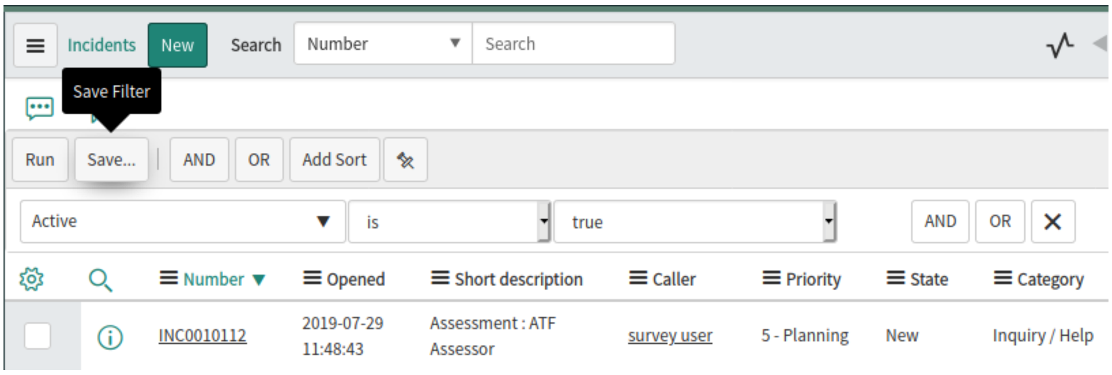
Number ▼	Opened	Short description	Caller	Priority	State
INC0010112	2019-07-29 11:48:43	Assessment : ATF Assessor	survey user	5 - Planning	New
INC0010111	2019-07-22 14:04:57	ATF : Test1	System Administrator	5 - Planning	New
INC0009009	2018-08-30 01:06:16	Unable to access the shared folder.	David Miller	4 - Low	New
INC0009005	2018-08-31 21:35:21	Email server is down.	David Miller	● 1 - Critical	New
INC0009004	2018-09-01 06:13:30	Defect tracking tool is down.	David Miller	3 - Moderate	Closed
INC0009003	2018-08-30 02:17:32	Cannot sign into the company portal app	David Miller	3 - Moderate	Closed
INC0009002	2018-09-16 05:49:23	My computer is not detecting the headphone device	David Miller	3 - Moderate	Closed

2. Show the filter bar.



Number ▼	Opened	Short description	Caller	Priority	State
INC0010112	2019-07-29 11:48:43	Assessment : ATF Assessor	survey user	5 - Planning	New

3. Adjust the filter options.
4. Save the filter.



Active is true

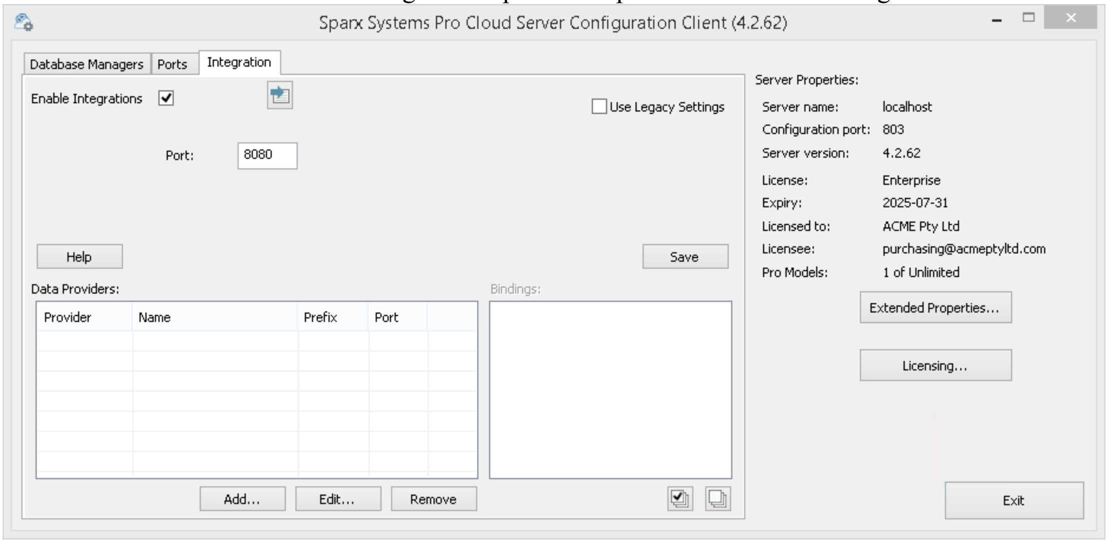
Number	Opened	Short description	Caller	Priority	State	Category
INC0010112	2019-07-29 11:48:43	Assessment : ATF Assessor	survey user	5 - Planning	New	Inquiry / Help

5. This filter (and the items it returns) will now be accessible to the Pro Cloud Server's ServiceNow integration.

Identify ServiceNow to Pro Cloud Server and Enterprise Architect

Work through these steps:

1. Double-click on the 'Pro Cloud Config' Desktop icon to open the Pro Cloud Config Client window.



Enable Integrations ☑

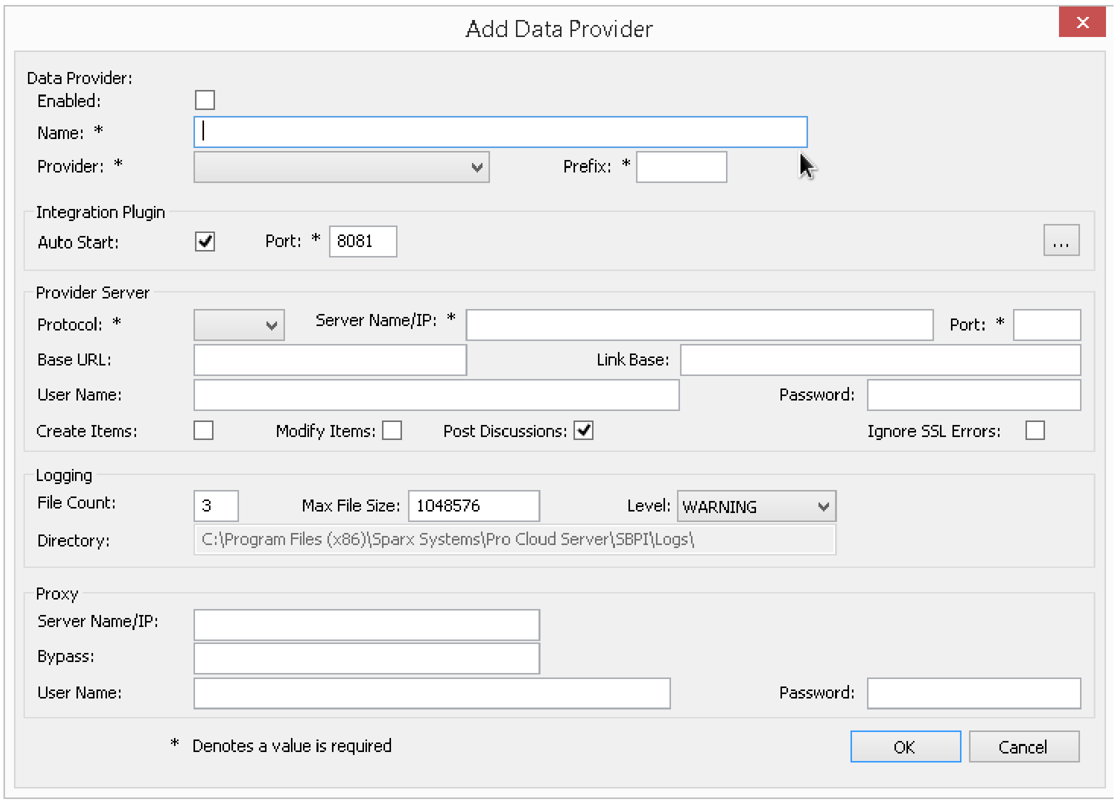
Use Legacy Settings ☐

Port: 8080

Server Properties:

Server name:	localhost
Configuration port:	803
Server version:	4.2.62
License:	Enterprise
Expiry:	2025-07-31
Licensed to:	ACME Pty Ltd
Licensee:	purchasing@acmeptyltd.com
Pro Models:	1 of Unlimited

2. Select the 'Integration' tab and select the 'Enable Integrations' checkbox. In the 'Port' field type the number of the Port the Integration Server is listening on for requests from the Pro Cloud Server.
3. Click on the Add button. The 'Add Data Provider' dialog displays.


Add Data Provider

Data Provider:
Enabled: ☐
Name: *
Provider: * Prefix: *

Integration Plugin
Auto Start: ✓ Port: * 8081

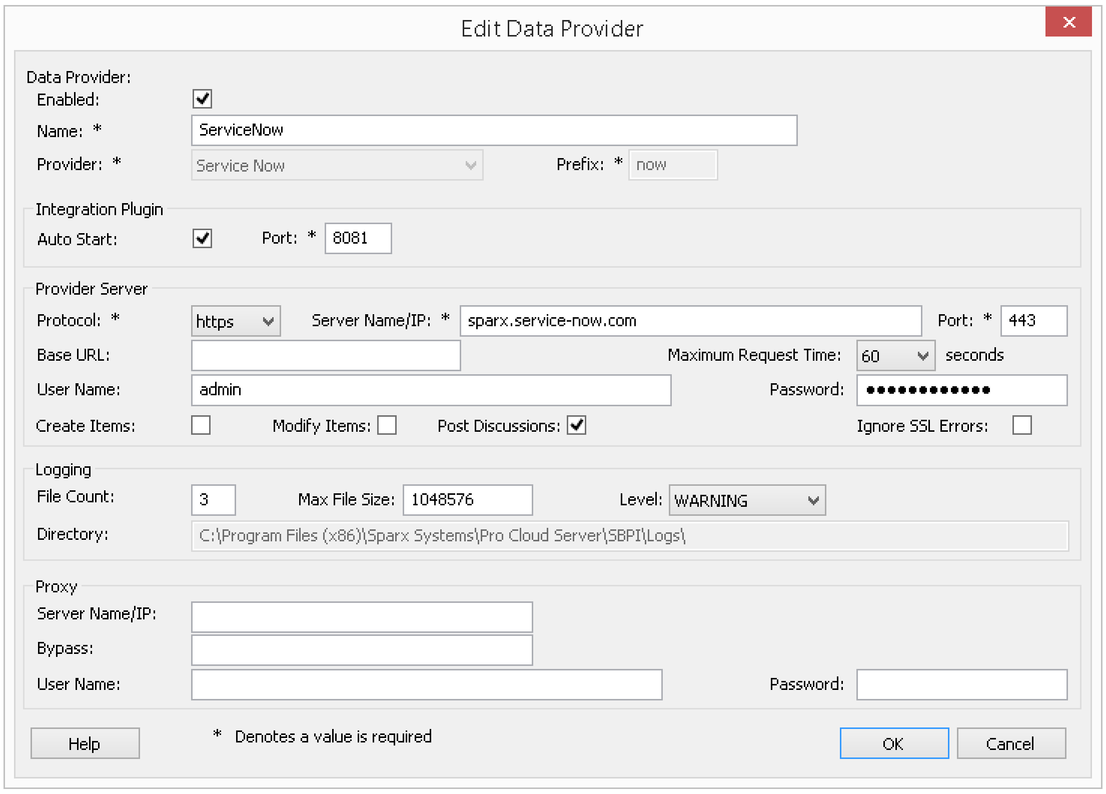
Provider Server
Protocol: * Server Name/IP: * Port: *
Base URL: Link Base:
User Name: Password:
Create Items: ☐ Modify Items: ☐ Post Discussions: ✓ Ignore SSL Errors: ☐

Logging
File Count: 3 Max File Size: 1048576 Level: WARNING
Directory: C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\Logs\

Proxy
Server Name/IP:
Bypass:
User Name: Password:

* Denotes a value is required

4. Select the 'Enabled' checkbox.
5. In the 'Name' field, type an appropriate connection name, such as 'ServiceNow'.
6. In the 'Provider' field, click on the drop-down arrow and select 'Service Now'; this automatically adds 'now' to the 'Prefix' field as well.
7. In the 'Integration Plugin' panel, in the 'Port' field, type the number of the Port that the Integration Server will use to communicate with the ServiceNow Plug-in.
8. In the 'Protocol' field, click on the drop-down arrow and select 'https'.
9. In the 'Server Name/IP' field, type the server name or IP address that the ServiceNow Plug-in will use to communicate with ServiceNow.
10. In the 'Port' field enter '443'.
11. Leave the 'Base URL' field blank for a default ServiceNow configuration. The field might be required for non-default ServiceNow Server configurations.
    Note that the fields in steps 8 to 11 are concatenated to make a web address; that is:
    <protocol>://<server>:<port>/baseURL (*baseURL* included if the field is not blank).
12. If you leave the 'User' and 'Password' fields blank, then each Enterprise Architect user will be prompted for their personal ServiceNow credentials, which can give a better usability.
13. If you prefer to set values in the 'User Name' and 'Password' fields, the values will be used in combination to access the data within ServiceNow. All current Enterprise Architect users of the ServiceNow installation will use the same set of credentials to read the external data.
14. Select the 'Create Items' and/or 'Modify Items' checkboxes as necessary, to allow users of the ServiceNow Plug-in to create and/or update items in ServiceNow using Enterprise Architect.
15. Leave the 'Ignore SSL Errors' option unchecked.



Edit Data Provider

Data Provider:

Enabled:	✓
Name: *	ServiceNow
Provider: *	Service Now
Prefix: *	now

Integration Plugin

Auto Start:	✓
Port: *	8081

Provider Server

Protocol: *	https
Server Name/IP: *	sparx.service-now.com
Port: *	443
Base URL:	
Maximum Request Time:	60 seconds
User Name:	admin
Password:	●●●●●●●●●●●●
Create Items:	☐
Modify Items:	☐
Post Discussions:	✓
Ignore SSL Errors:	☐

Logging

File Count:	3
Max File Size:	1048576
Level:	WARNING
Directory:	C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\Logs\

Proxy

Server Name/IP:	
Bypass:	
User Name:	
Password:	

* Denotes a value is required

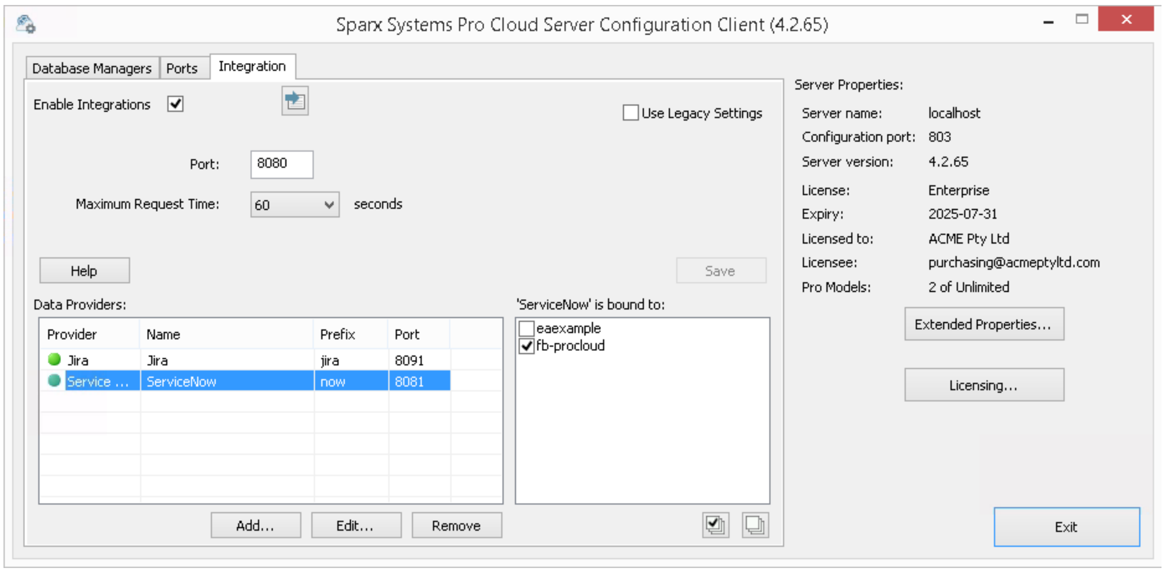
16. You can leave other fields blank or set to their default values. Click on the OK button to complete the configuration. This returns you to the 'Integration' tab of the Pro Cloud Server Configuration Client window.

Identify Enterprise Architect models to ServiceNow

On the 'Integration' tab of the Pro Cloud Server Configuration Client window, you will now see:

- On the lower left of the screen, the 'Data Providers' panel listing ServiceNow as a Data Provider
- On the lower right of the screen, the ''ServiceNow' is bound to:' panel displaying a check list of the Enterprise Architect models from your Pro-enabled Database Manager(s)

You can either click on the checkbox against each Enterprise Architect model you want to bind to ServiceNow, or click on the to select all of them at once.



Sparx Systems Pro Cloud Server Configuration Client (4.2.65)

Enable Integrations ✓

Use Legacy Settings ☐

Port: 8080

Maximum Request Time: 60 seconds

Data Providers:

Provider	Name	Prefix	Port
● Jira	Jira	jira	8091
● Service ...	ServiceNow	now	8081

'ServiceNow' is bound to:

☐ eaexample
✓ fb-procloud

Server Properties:

Server name:	localhost
Configuration port:	803
Server version:	4.2.65
License:	Enterprise
Expiry:	2025-07-31
Licensed to:	ACME Pty Ltd
Licensee:	purchasing@acmeptyltd.com
Pro Models:	2 of Unlimited

Click on the Exit button.

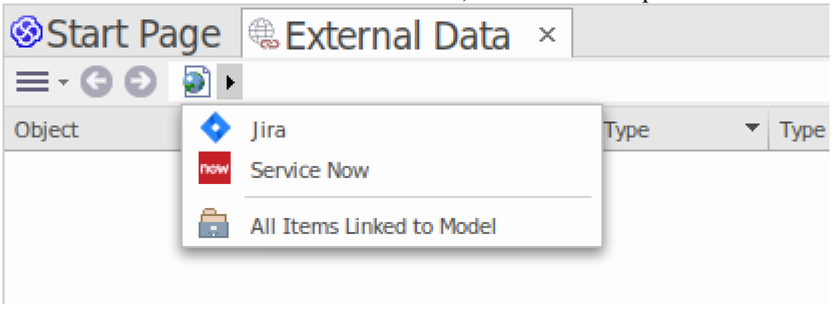
Restart the PCS Service to apply the changes you have made.

You can now go into one of your Enterprise Architect models and check the integration with ServiceNow.

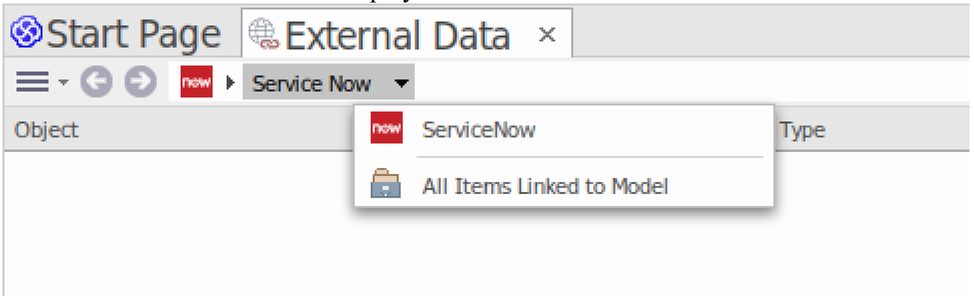
Test the Integration

In Enterprise Architect open one of the models you have identified as being bound to ServiceNow.

17. Select the ribbon option 'Specialize > Tools > System Integration > Open External Data'.
18. In the toolbar of the External Data window, click on the drop-down arrow to the right of the 'globe' icon.



19. Select 'Service Now' from the list; this adds the provider to the breadcrumb trail in the toolbar.
20. Click on 'ServiceNow' in the breadcrumb trail to display a list of ServiceNow providers that have been configured and bound to this model. If nothing is listed then no provider has been bound to this model.
21. Click the Provider that was set up and bound to this model, to display a list of ServiceNow tables that are now available for selection to work with in this Enterprise Architect model.
22. Select the ServiceNow table to display its list of User filters.



If nothing shows here then you might not have sufficient privileges to view the tables. However, if you have not had any User filters created for you the list just contains '[ ]' or an information message.

(If you cannot see ServiceNow or a list of ServiceNow tables, check through the procedures in this topic, or the broader and more detailed topics in the Integration Plug-ins section to see if there is a step you have missed or a prerequisite that is not satisfied. You can also check the Troubleshooting Help topic.)

Data Mapping

In order for ServiceNow and Enterprise Architect to smoothly exchange data items, you have to review and - where necessary - edit:

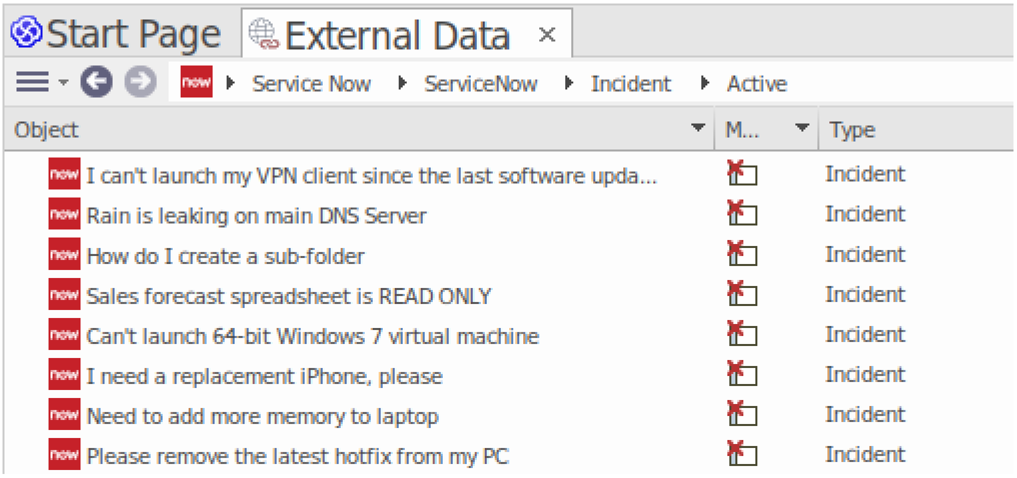
- What type of ServiceNow item corresponds to which type of Enterprise Architect element, and
- What property of a ServiceNow item corresponds to which property of an Enterprise Architect element

These tasks are fully explained in the Configuration Help topic.

Review ServiceNow Data

When you tested the communication between Enterprise Architect and ServiceNow, on the External Data window, you opened the breadcrumb trail to show a list of ServiceNow projects.

When you select one of the projects, you again click on the drop-down arrow and select one of the filters from the list, which then lists the items from that filter in the body of the window.



Object	M...	Type
I can't launch my VPN client since the last software upda...		Incident
Rain is leaking on main DNS Server		Incident
How do I create a sub-folder		Incident
Sales forecast spreadsheet is READ ONLY		Incident
Can't launch 64-bit Windows 7 virtual machine		Incident
I need a replacement iPhone, please		Incident
Need to add more memory to laptop		Incident
Please remove the latest hotfix from my PC		Incident

If this list is too long, you can click on the 'Hamburger' icon in the window toolbar and select options such as:

- 'Linked Items' to show only those ServiceNow items that are linked to Enterprise Architect elements
- 'Items Linked to Current Context' to show only the ServiceNow items that are linked to the currently selected Enterprise Architect element or
- 'Unlinked items' to show only those ServiceNow items that have not yet been linked to Enterprise Architect elements

Select the appropriate option, if necessary, and then explore the content of selected ServiceNow items. Each of the Properties window, Notes window and Inspector window have separate tabs or versions of the window to display any properties, notes and features (respectively) of the selected item. These are illustrated in the External Item Details Help topic.

Working with ServiceNow items and Enterprise Architect elements

Having created a working communication between Enterprise Architect and ServiceNow, you can:

- Create a new element in Enterprise Architect linked to a ServiceNow item
- Create a new ServiceNow item linked to an element in Enterprise Architect
- Link an existing element in Enterprise Architect to a ServiceNow item
- Update the ServiceNow item with changes to the Enterprise Architect element
- Update the Enterprise Architect element with changes to the ServiceNow item
- Update all linked Enterprise Architect elements with any changes to the ServiceNow items

These actions are all explained in the Linking Items Help topic.

Walkthrough: Jira Integration

This walkthrough helps you to set up and use the integration capability of Pro Cloud Server to integrate your Enterprise Architect models with the corresponding Jira Software projects, via the Cloud installation option.

Using the integration with Jira you can:

- Link Enterprise Architect elements with their counterparts in Jira
- Create Enterprise Architect elements based on Jira items, and vice-versa
- Synchronize changes between Enterprise Architect and Jira, as and when needed
- Perform a traceability analysis
- Add comments to Jira items from Enterprise Architect

When Pro Cloud Server is installed on your system, it provides a prompt to also install the optional Integration Plug-ins component; by default, the component is installed into the C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\ folder (assuming a 64 bit machine is being used). The Integration Plug-ins component includes the:

- Integration (SBPI) Server (sbpi.exe)
- The plug-in .exe file for each of the many integrated tools, including the one for Jira (JiraSbpi.exe)

The JiraSbpi.exe plug-in interacts with Atlassian's issue tracking system. It is able to list a user's favorite filters (also known as *starred* filters). Each filter will then list all the Jira items returned by the filter. These filters are configured in Jira in a Favorite folder for each user logging in, prior to Enterprise Architect requesting that the Integration Plug-in should connect to them. You can view only those Jira items that are part of a filter, so create appropriate filters in Jira. For example, if you want to link Enterprise Architect Requirements to Jira User Stories, create a filter that will show the relevant User Stories.

Jira itself has two installation options, Jira Cloud and Jira Server. The Jira Server option is being phased out, so these procedures assume that you are working with Jira Cloud. However, if you are working with Jira Server, there are some comments in the procedures to accommodate that.

Access Tokens

Where Jira is hosted on an Atlassian server and not hosted locally, it requires using a TokenID that is created on the Atlassian site, for a single predefined user set for logging into Atlassian. See the Atlassian web site pages on Access Tokens - a general review (https://www.atlassian.com/software/access/guide/elements/api-token-controls#what-are-api-token-controls) and a discussion on creating tokens (https://confluence.atlassian.com/cloud/api-tokens-938839638.html). See the links under *Learn More* below.

Prerequisites

In order to set up integration between Jira and Enterprise Architect, you must first:

- Be using a licensed installation of Pro Cloud Server with the Integration Plug-ins component, as just discussed
- Select the 'Enable Pro Features (OSLC, WebEA and Integration)' option in the Pro Cloud Server configuration for each Database Manager System hosting your Enterprise Architect models
- Have physical network access between the PCS server and the Jira server hosting the external Jira installation and its data
- Have user credentials to access the external data
- Have a 'Favorite' folder of appropriate filters set up in Jira, as just discussed
- Be using Enterprise Architect Corporate, Unified or Ultimate edition v14 or later

We will assume that:

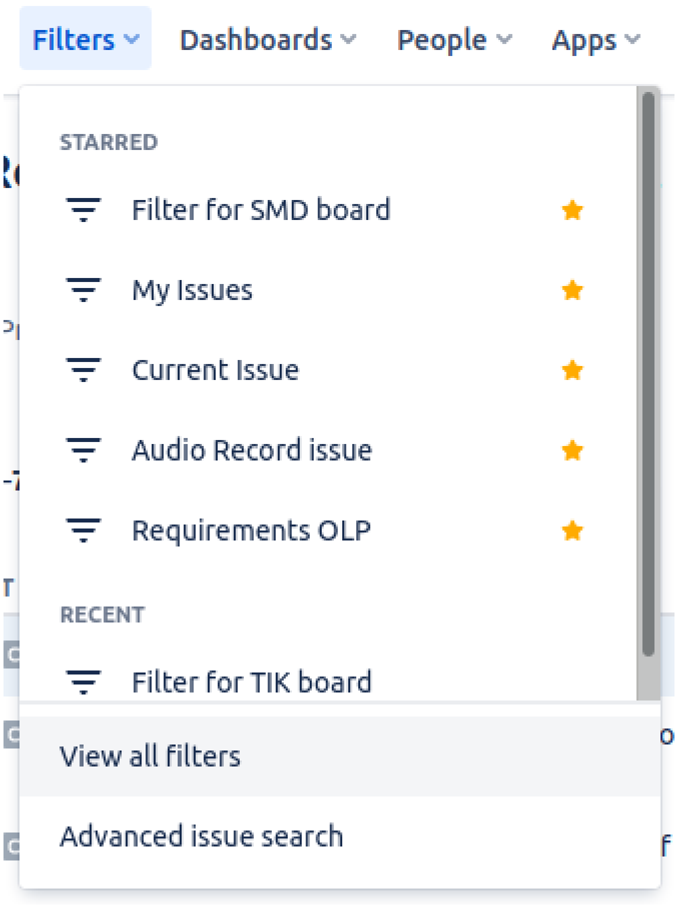
- The Integration Server and the Jira Integration Plug-in are to run on the same machine as the Pro Cloud Server
- You are defining the Integration Server options using the 'Integration' tab in the Pro Cloud Server Configuration Client rather than by manually manipulating the registry or configuration files or by using the WebConfig interface

Define starred filters in Jira

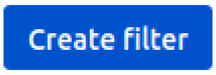
The Pro Cloud Server's Jira integration allows access to items which are returned by Jira's starred filters. Before using the integration you should ensure some starred filters have been set up to return the items that you want to be accessible in Enterprise Architect.

To define starred filters in Jira, follow these steps:

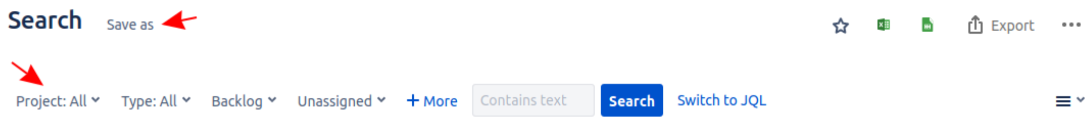
23. Within Jira, select the 'Filters' dropdown menu, then 'View All Filters'.



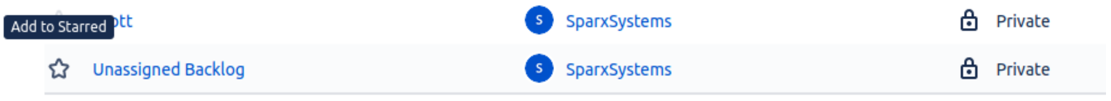
24. At the top right of the filters list, click on the 'Create Filter' button.



25. Adjust the filter/search settings (such as Project and Type) then use the 'Save as' option to save a new filter.



26. Now return to the list of all filters ('Filters | View All Filters'). In the filter list you can use the star icon to add the filter to your starred filters.

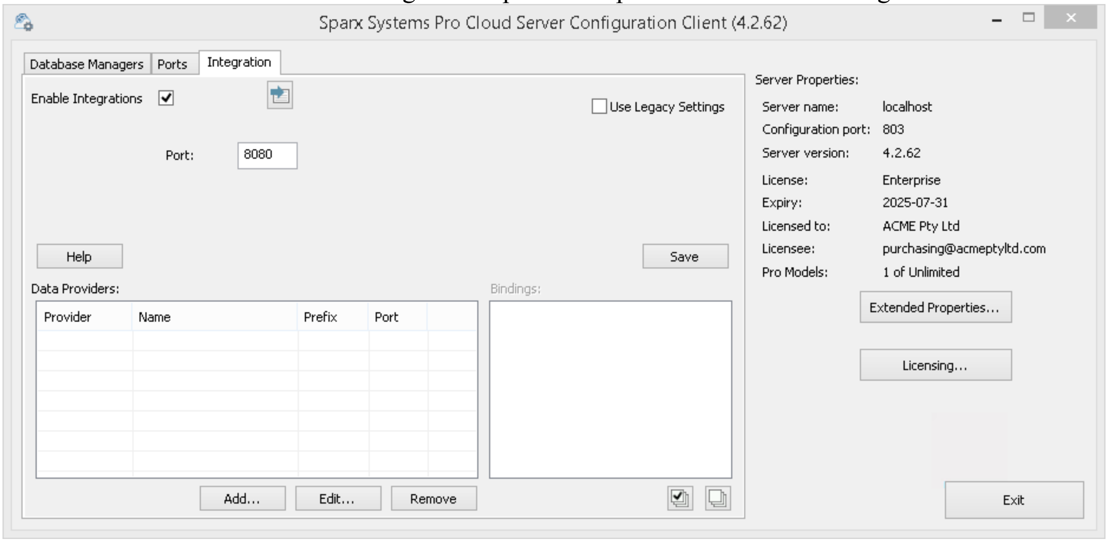


27. This filter (and items it returns) should now be accessible to the Pro Cloud Server's Jira integration

Identify Jira to Pro Cloud Server and Enterprise Architect

Work through these steps:

28. Double-click on the 'Pro Cloud Config' Desktop icon to open the Pro Cloud Config Client window.



Sparx Systems Pro Cloud Server Configuration Client (4.2.62)

Database Managers | Ports | Integration

Enable Integrations ☑

☐ Use Legacy Settings

Port: 8080

Help Save

Data Providers:

Provider	Name	Prefix	Port

Bindings:

Add... Edit... Remove

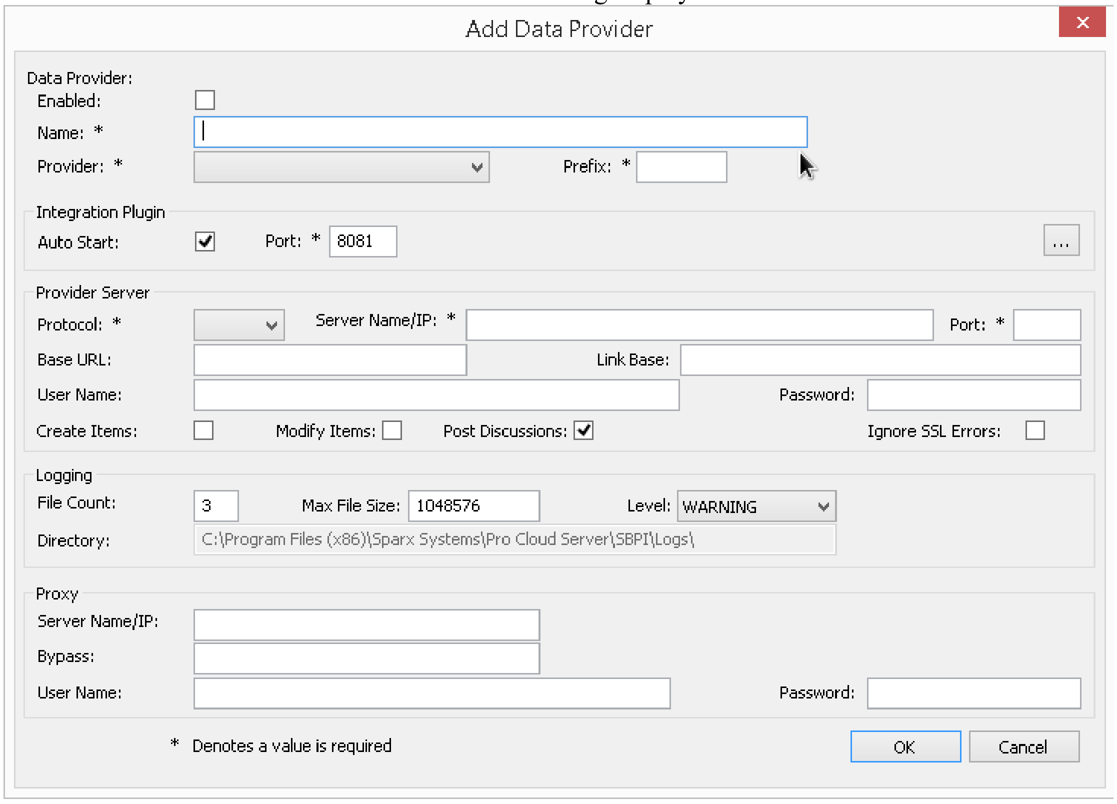
Server Properties:

Server name:	localhost
Configuration port:	803
Server version:	4.2.62
License:	Enterprise
Expiry:	2025-07-31
Licensed to:	ACME Pty Ltd
Licensee:	purchasing@acmeptyltd.com
Pro Models:	1 of Unlimited

Extended Properties... Licensing... Exit

29. Select the 'Integration' tab and select the 'Enable Integrations' checkbox. In the 'Port' field type the number of the Port the Integration Server is listening on for requests from the Pro Cloud Server.

30. Click on the Add button. The 'Add Data Provider' dialog displays.



Add Data Provider

Data Provider:
Enabled: ☐
Name: *
Provider: * Prefix: *

Integration Plugin
Auto Start: ☑ Port: * 8081 ...

Provider Server
Protocol: * Server Name/IP: * Port: *
Base URL: Link Base:
User Name: Password:
Create Items: ☐ Modify Items: ☐ Post Discussions: ☑ Ignore SSL Errors: ☐

Logging
File Count: 3 Max File Size: 1048576 Level: WARNING
Directory: C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\Logs\

Proxy
Server Name/IP:
Bypass:
User Name: Password:

* Denotes a value is required

OK Cancel

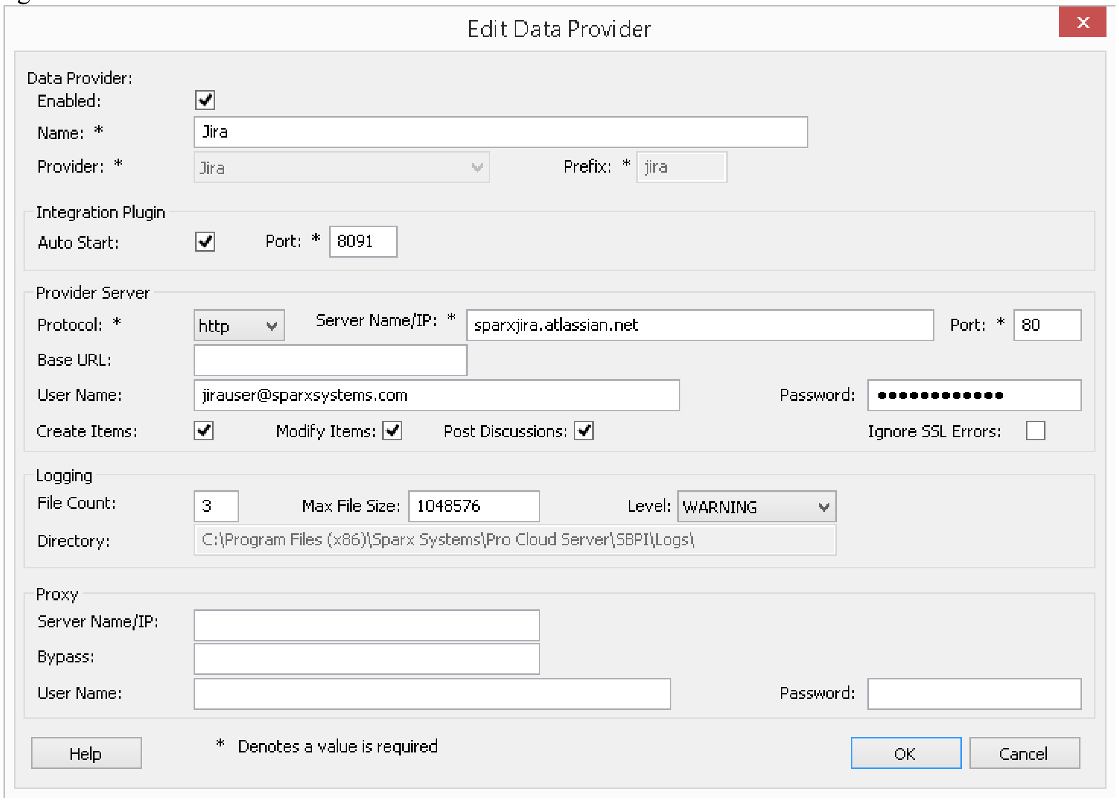
31. Select the 'Enabled' checkbox.

32. In the 'Name' field, type an appropriate connection name, such as 'Jira'.

33. In the 'Provider' field, click on the drop-down arrow and select 'Jira'; this automatically adds 'Jira' to the 'Prefix' field

as well.

34. In the 'Integration Plugin' panel, in the 'Port' field, type the number of the Port that the Integration Server will use to communicate with the Jira Plug-in.
35. In the 'Protocol' field, click on the drop-down arrow and select the protocol that the Jira Plug-in will use to communicate with Jira.
    - For a Jira Cloud default installation, this must be 'HTTPS'
    - For a Jira Server installation,the default is 'HTTP'
36. In the 'Server Name/IP' field, type the server name or IP address that the Jira Plug-in will use to communicate with Jira.
    - For a Jira Cloud default installation, this must be <account>.atlassian.net
    - For a Jira Server installation, the default is <name of server or host>
37. In the 'Port' field type the number of the Port that the Jira Plug-in will use to communicate with Jira.
    - For a Jira Cloud default installation, this must be '443'
    - For a Jira Server installation,the default is '8080'
38. Leave the 'Base URL' field blank for Jira Cloud, and for a default Jira Server installation. The field might be required for non-default Jira Server configurations.
    Note that the fields in steps 8 to 11 are concatenated to make a web address; that is:
    <protocol>://<server>:<port>/baseURL (*baseURL* included if the field is not blank).
39. If you leave the 'User' and 'Password' fields blank, then each Enterprise Architect user will be prompted for their personal Jira credentials, which can give a better usability.
40. If you prefer to set values in the 'User Name' and 'Password' fields, the values will be used in combination to access the data within Jira. All current Enterprise Architect users of the Jira installation will use the same set of credentials to read the external data.
41. Select the 'Create Items', 'Modify Items' and/or 'Post Discussions' checkboxes as necessary, to allow users of the Jira Plug-in to create and/or update items and/or create Discussion posts in Jira using Enterprise Architect.
42. If you want the Jira Plug-in to ignore SSL-related errors that occur as a result of communicating with Jira, select the 'Ignore SSL Errors' checkbox.


Edit Data Provider

Data Provider:
Enabled: ✓
Name: * Jira
Provider: * Jira
Prefix: * jira

Integration Plugin
Auto Start: ✓
Port: * 8091

Provider Server
Protocol: * http
Server Name/IP: * sparxjira.atlassian.net
Port: * 80
Base URL:
User Name: jirauser@sparxsystems.com
Password: ●●●●●●●●●●●●●
Create Items: ✓
Modify Items: ✓
Post Discussions: ✓
Ignore SSL Errors:

Logging
File Count: 3
Max File Size: 1048576
Level: WARNING
Directory: C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\Logs\

Proxy
Server Name/IP:
Bypass:
User Name:
Password:

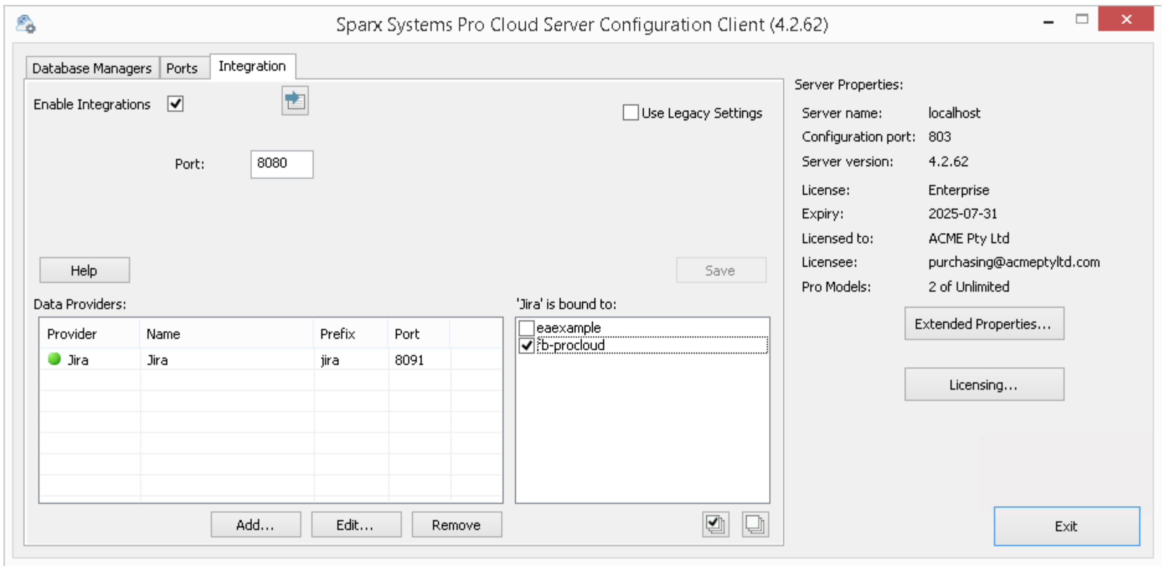
43. You can leave other fields blank or set to their default values. Click on the OK button to complete the configuration. This returns you to the 'Integration' tab of the Pro Cloud Server Configuration Client window.

Identify Enterprise Architect models to Jira

On the 'Integration' tab of the Pro Cloud Server Configuration Client window, you will now see:

- On the lower left of the screen, the 'Data Providers' panel listing Jira as a data Provider
- On the lower right of the screen, the "Jira' is bound to:' panel displaying a check list of the Enterprise Architect models from your Pro-enabled Database Manager(s)

You can either click on the checkbox against each Enterprise Architect model you want to bind to Jira, or click on the ☑ to select all of them at once.



Sparx Systems Pro Cloud Server Configuration Client (4.2.62)

Database Managers | Ports | Integration

Enable Integrations ☑

☐ Use Legacy Settings

Port: 8080

Help   Save

Data Providers:

Provider	Name	Prefix	Port
● Jira	Jira	jira	8091

Add... Edit... Remove

'Jira' is bound to:

☐ eaexample
☑ fb-procloud

Server Properties:

Server name:	localhost
Configuration port:	803
Server version:	4.2.62
License:	Enterprise
Expiry:	2025-07-31
Licensed to:	ACME Pty Ltd
Licensee:	purchasing@acmeptyltd.com
Pro Models:	2 of Unlimited

Extended Properties...

Licensing...

Exit

Click on the Exit button.

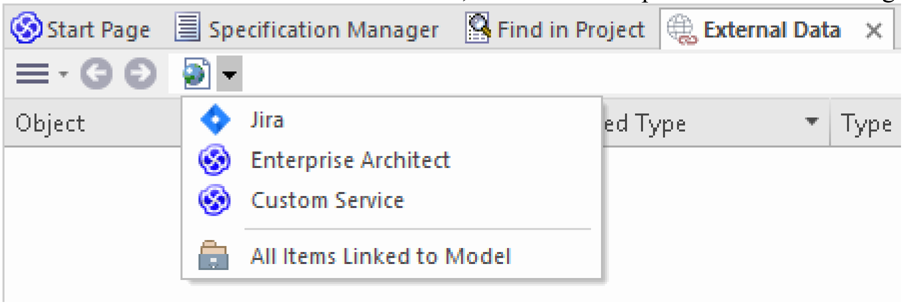
Restart the PCS Service to apply the changes you have made.

You can now go into one of your Enterprise Architect models and check the integration with Jira.

Test the Integration

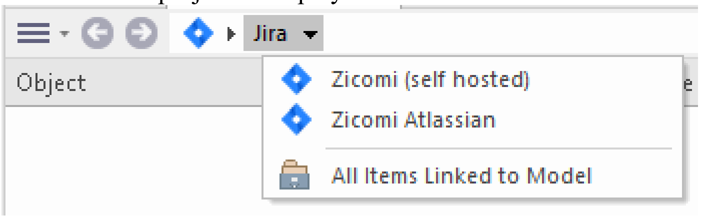
In Enterprise Architect open one of the models you have selected in the list: *'Jira' is bound to. S*ee the image above.

1. Select the ribbon option Specialize > Tools > System Integration > Open External Data.
2. In the toolbar of the External Data window, click on the drop-down arrow to the right of the 'globe' icon.



3. Select 'Jira' from the list; this adds the provider to the breadcrumb trail in the toolbar.
4. Click on 'Jira' in the breadcrumb trail to display a list of Jira providers that have been configured and bound to this model. If nothing is listed then no provider has been bound to this model.
5. Click the Provider that was set up and bound to this model, to display a list of Jira projects that are now available for selection to work with in this Enterprise Architect model.

6. Select the Jira project to display a list of User filters.



If nothing shows here then you might not have sufficient privileges to view the projects. However, if you have not had any User filters created for you the list just contains '[ ]' or an information message.

(If you cannot see Jira or a list of Jira projects, check through the procedures in this topic, or the broader and more detailed topics in the Integration Plug-ins section to see if there is a step you have missed or a prerequisite that is not satisfied. You can also check the Troubleshooting Help topic.)

Data Mapping

In order for Jira and Enterprise Architect to smoothly exchange data items, you have to review and - where necessary - edit:

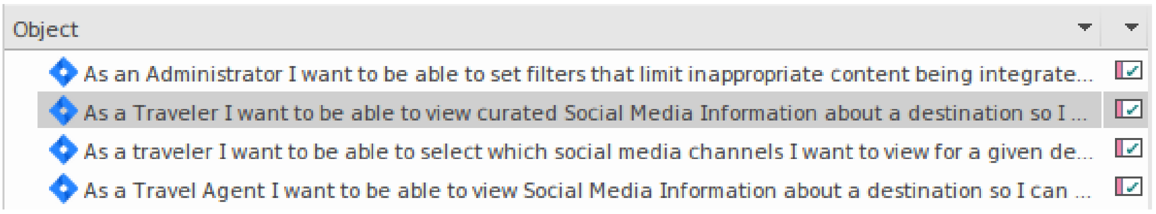
- What type of Jira item corresponds to which type of Enterprise Architect element, and
- What property of a Jira item corresponds to which property of an Enterprise Architect element.

These tasks are fully explained in the Configuration Help topic, which uses Jira as the example Data Provider.

Review Jira Data

When you tested the communication between Enterprise Architect and Jira, on the External Data window, you opened the breadcrumb trail to show a list of Jira projects.

When you select one of the projects, you again click on the drop-down arrow and select one of the filters from the list, which then lists the items from that filter in the body of the window.


Object
As an Administrator I want to be able to set filters that limit inappropriate content being integrate...
As a Traveler I want to be able to view curated Social Media Information about a destination so I ...
As a traveler I want to be able to select which social media channels I want to view for a given de...
As a Travel Agent I want to be able to view Social Media Information about a destination so I can ...

If this list is too long, you can click on the 'Hamburger' icon in the window toolbar and select options such as:

- 'Linked Items' to show only those Jira items that are linked to Enterprise Architect elements
- 'Items Linked to Current Context' to show only the Jira items that are linked to the currently selected Enterprise Architect element or
- 'Unlinked items' to show only those Jira items that have not yet been linked to Enterprise Architect elements

Select the appropriate option, if necessary, and then explore the content of selected Jira items. Each of the Properties window, Notes window, Inspector window and Discuss tab of the Discuss & Review window have separate tabs or versions of the window to display any properties, notes, features and discussions (respectively) of the selected item. These are illustrated in the External Item Details Help topic.

Working with Jira items and Enterprise Architect elements

Having created a working communication between Enterprise Architect and Jira, you can:

- Create a new element in Enterprise Architect linked to a Jira item
- Create a new Jira item linked to an element in Enterprise Architect
- Link an existing element in Enterprise Architect to a Jira item
- Update the Jira item with changes to the Enterprise Architect element
- Update the Enterprise Architect element with changes to the Jira item
- Update all linked Enterprise Architect elements with any changes to the Jira items

These actions are all explained in the Linking Items Help topic, which uses Jira as the example Data Provider.

Walkthrough: DevOps Integration

This walkthrough helps you to set up and use the Pro Cloud Server's DevOps/Team Foundation Server integration.

Note, in 2019 Microsoft's Team Foundation Server was renamed to Azure DevOps. More specifically, the cloud hosted Visual Studio Team Services (VSTS) was renamed to Azure DevOps Services, and on-premises Visual Studio Team Foundation Server (TFS) was renamed to Azure DevOps Server.

The Pro Cloud Server integration is compatible with Azure DevOps (both cloud hosted and on-premises) and Team Foundation Server (TFS). This walkthrough will primarily use the current name 'DevOps', however within the Pro Cloud Server and Enterprise Architect, the plugin will often be referred to as Team Foundation Server or TFS.

Using the integration with DevOps you can:

- Link Enterprise Architect elements with their counterparts in DevOps
- Create Enterprise Architect elements based on DevOps items
- Synchronize changes between Enterprise Architect and DevOps, as and when needed
- Perform a traceability analysis

Prerequisites

In order to set up integration between DevOps and Enterprise Architect, you must first:

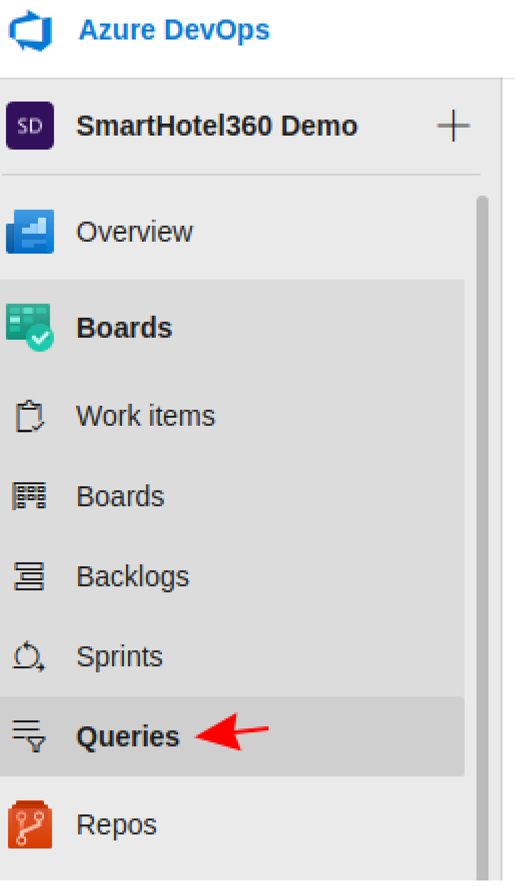
- Be using a licensed installation of Pro Cloud Server with the Integration Plug-ins component
- Select the 'Enable Pro Features (OSLC, WebEA and Integration)' option in the Pro Cloud Server configuration for each Database Manager System hosting your Enterprise Architect models
- Have network access between the PCS server and DevOps
- Have user credentials to access the external data
- Be using Enterprise Architect Corporate, Unified or Ultimate edition v14 or later

We will assume that the Integration Server and the DevOps Integration Plug-in are to run on the same machine as the Pro Cloud Server

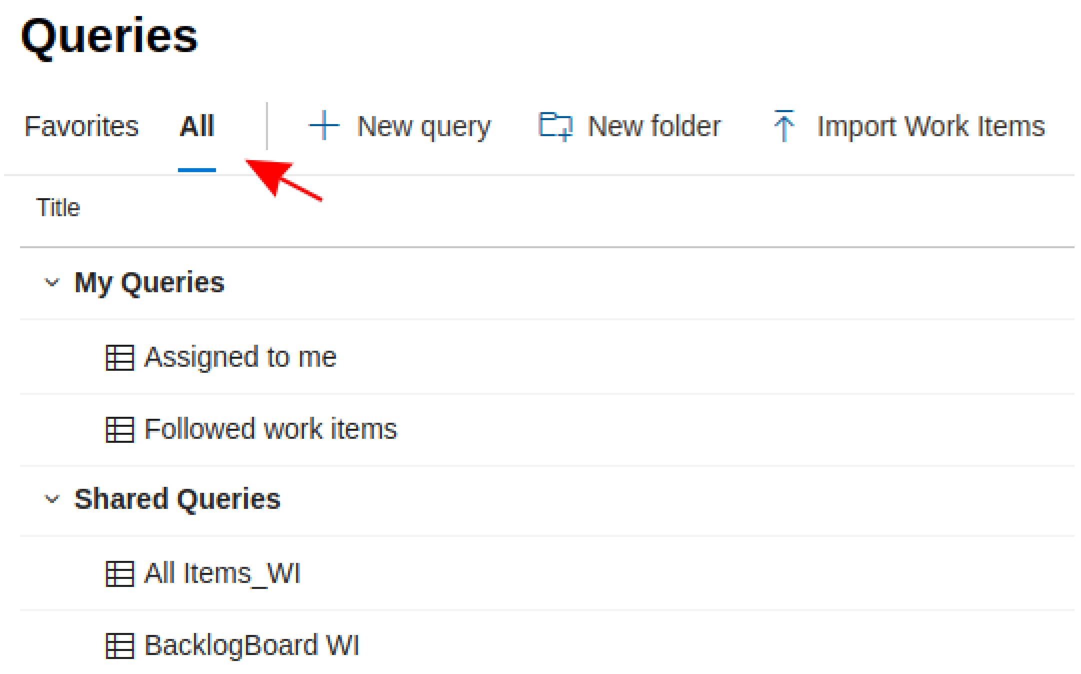
Define DevOps Queries

The DevOps integration allows access to work items via queries which have been defined in DevOps. You can view your existing queries and create new ones as described below.

7. Login to Azure DevOps and select your Project.
8. From the left panel select 'Boards', then 'Queries'.

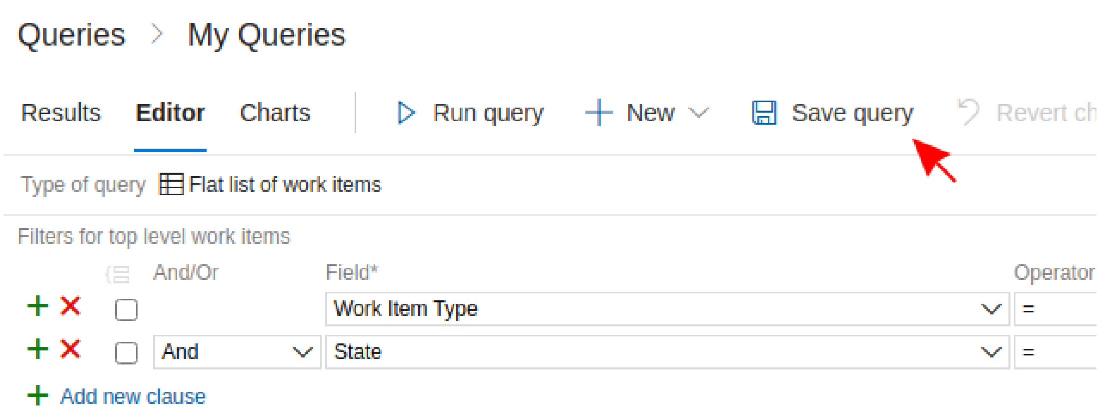


9. To view all queries which will be accessible to the DevOps integration click on 'All'. The Queries will be grouped into 'My Queries' and 'Shared Queries'.



10. A new query can be created using the 'New query' button.

11. After adjusting the query's filter options, click on the 'Save query' button to save the query.



Type of query: Flat list of work items

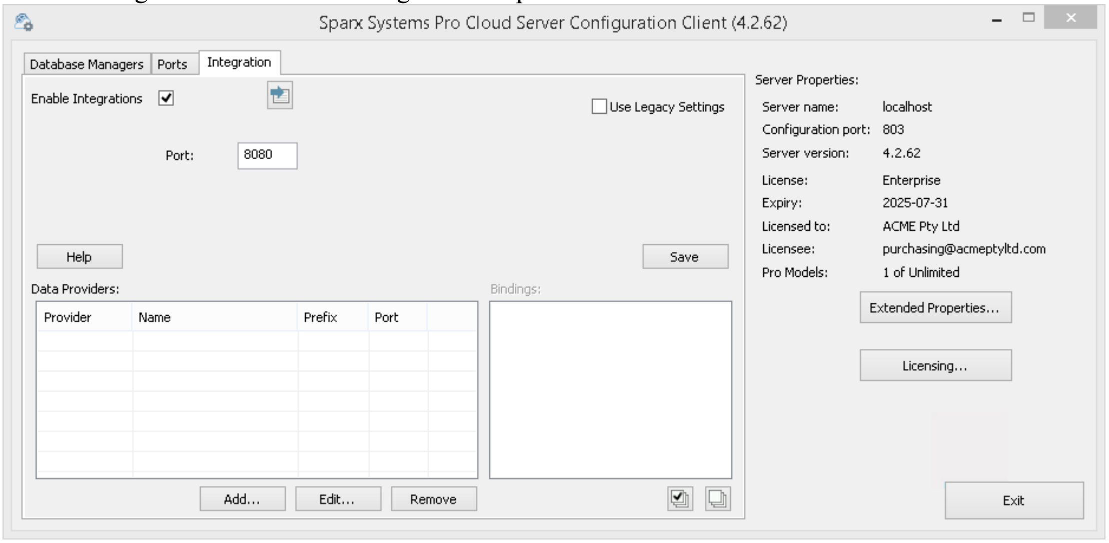
Filters for top level work items

	And/Or	Field*	Operator
☐		Work Item Type	=
☐	And	State	=

Identify DevOps to Pro Cloud Server and Enterprise Architect

Work through these steps:

12. Double-click on the 'Pro Cloud Config' Desktop icon to open the Pro Cloud Config Client window.
13. Select the 'Integration' tab and select the 'Enable Integrations' checkbox. In the 'Port' field type the number of the Port the Integration Server is listening on for requests from the Pro Cloud Server.



Enable Integrations ✓
Use Legacy Settings ☐
Port: 8080

Data Providers:

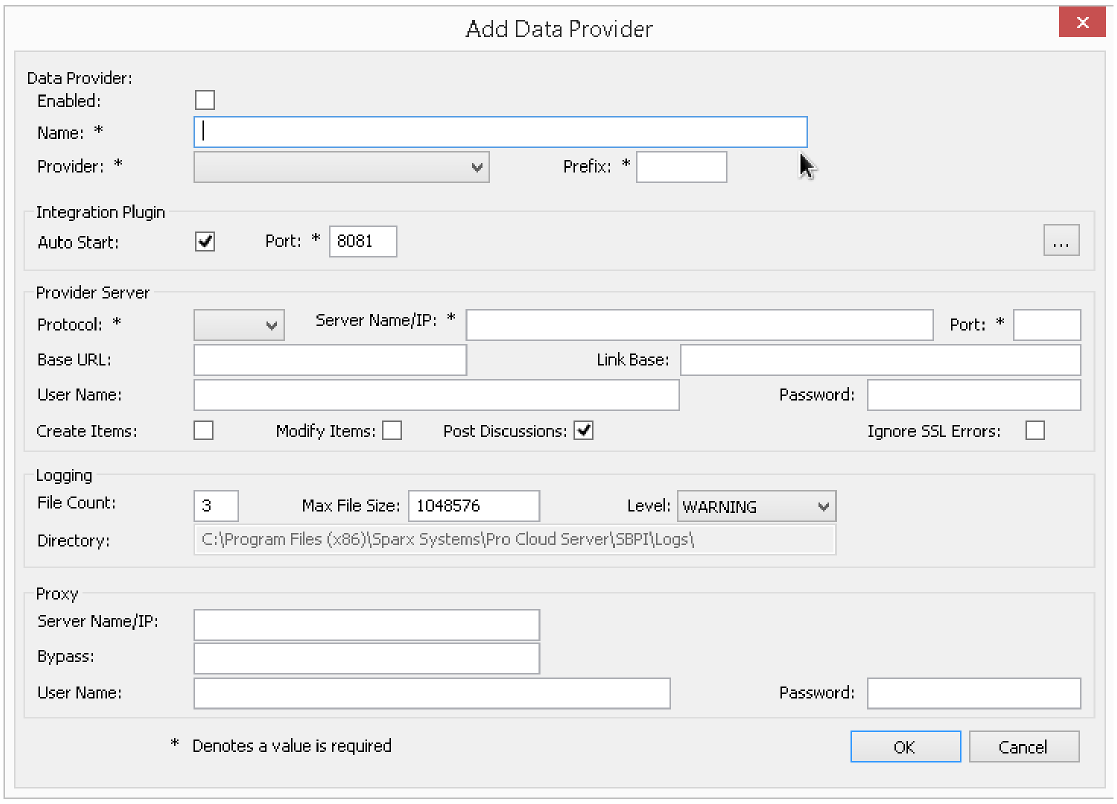
Provider	Name	Prefix	Port

Bindings:

Server Properties:

Server name:	localhost
Configuration port:	803
Server version:	4.2.62
License:	Enterprise
Expiry:	2025-07-31
Licensed to:	ACME Pty Ltd
Licensee:	purchasing@acmeptyltd.com
Pro Models:	1 of Unlimited

14. Click on the Add button. The 'Add Data Provider' dialog displays.


Add Data Provider

Data Provider:
Enabled: ☐
Name: *
Provider: * Prefix: *

Integration Plugin
Auto Start: ✓ Port: * 8081

Provider Server
Protocol: * Server Name/IP: * Port: *
Base URL: Link Base:
User Name: Password:
Create Items: ☐ Modify Items: ☐ Post Discussions: ✓ Ignore SSL Errors: ☐

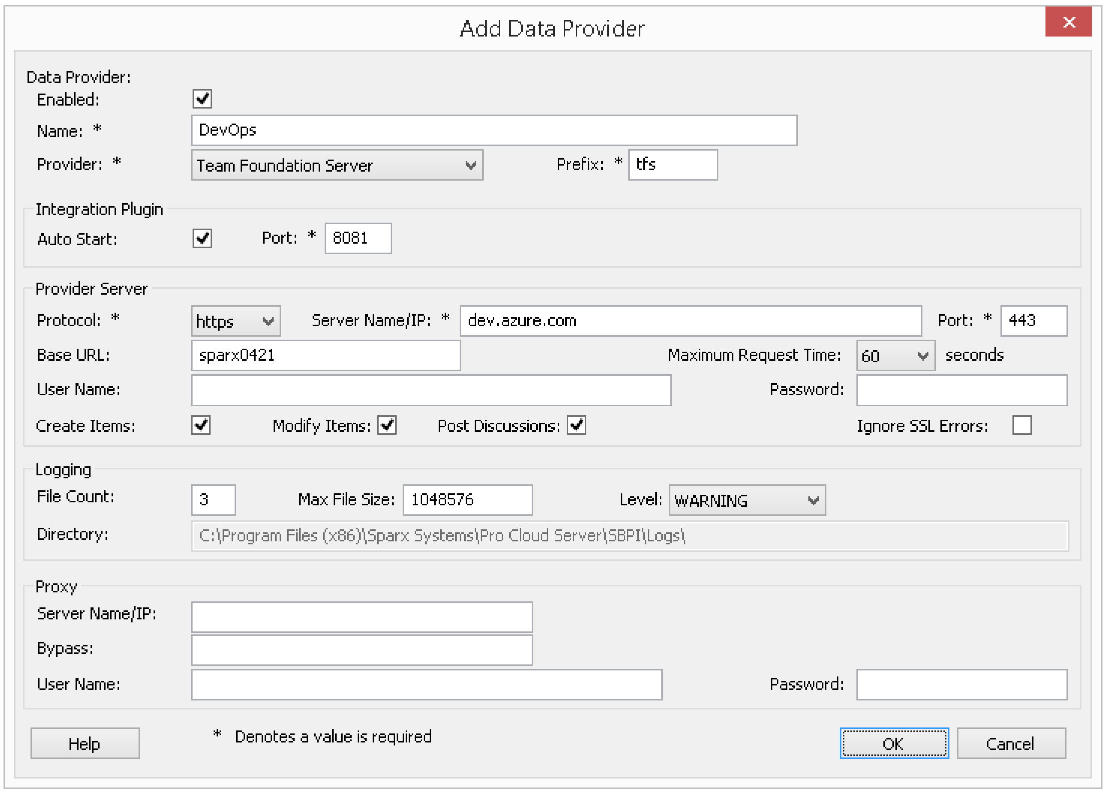
Logging
File Count: 3 Max File Size: 1048576 Level: WARNING
Directory: C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\Logs\

Proxy
Server Name/IP:
Bypass:
User Name: Password:

* Denotes a value is required

15. Select the 'Enabled' checkbox.
16. In the 'Name' field, type an appropriate connection name, such as 'DevOps'.
17. In the 'Provider' field, click on the drop-down arrow and select 'Team Foundation Server'; this automatically adds 'tfs' to the 'Prefix' field as well.
18. In the 'Integration Plugin' panel, in the 'Port' field, type the number of the Port that the Integration Server will use to communicate with the DevOps Plug-in.
19. If you are using the **Cloud-hosted 'Azure DevOps Services'**, fill in the Provider Server fields as shown:

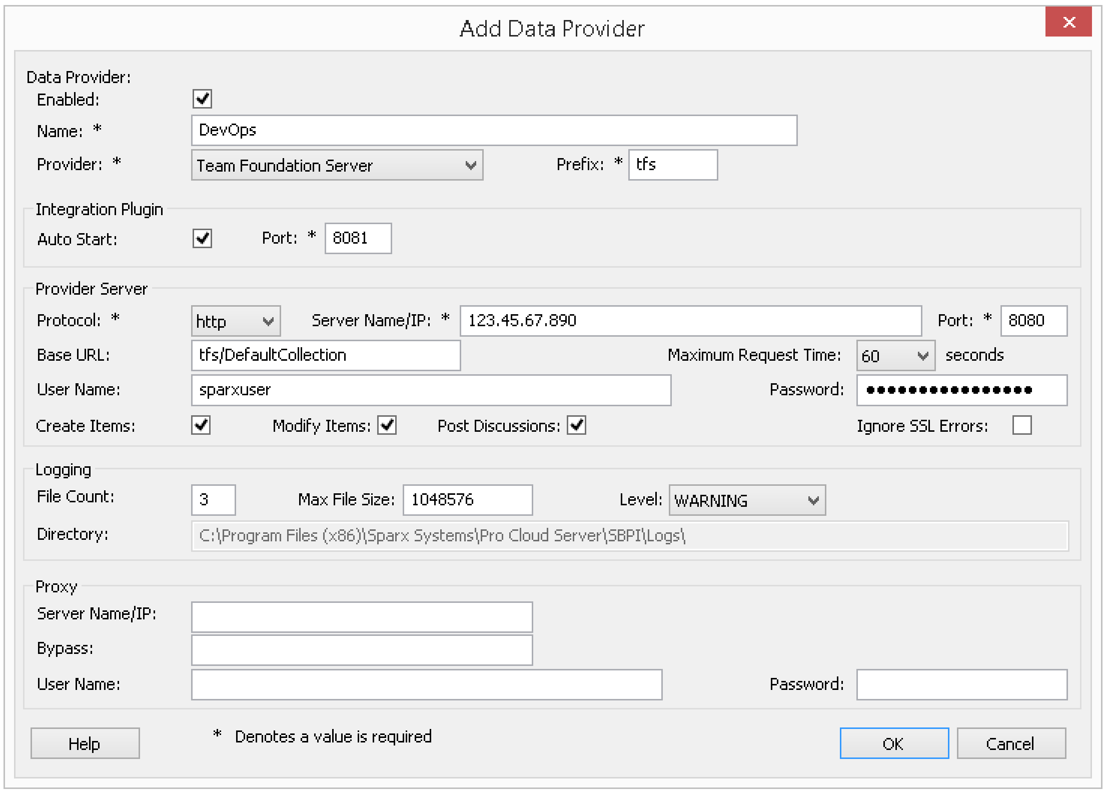
    Protocol: https
    Server Name/IP: dev.azure.com
    Port: 443
    Base URL: Enter the DevOps organization name. This can be found at the end of your DevOps URL. For example, If your url is https://dev.azure.com/org1/, then in the 'Base URL' field you would enter 'org1'.
    User Name: Leave this field empty. Users will be prompted for credentials when using the integration within Enterprise Architect.
    Password: Leave this field empty.
    Create Items / Modify Items / Post Discussions: Enable these checkboxes as necessary, to allow users of the DevOps Plug-in to create and/or update items and/or create Discussion posts in DevOps using Enterprise Architect.
    Ignore SSL Errors: Leave this option unchecked.



Data Provider:	
Enabled:	✓
Name: *	DevOps
Provider: *	Team Foundation Server
Prefix: *	tfs
Integration Plugin	
Auto Start:	✓
Port: *	8081
Provider Server	
Protocol: *	https
Server Name/IP: *	dev.azure.com
Port: *	443
Base URL:	sparx0421
Maximum Request Time:	60 seconds
User Name:	
Password:	
Create Items:	✓
Modify Items:	✓
Post Discussions:	✓
Ignore SSL Errors:	
Logging	
File Count:	3
Max File Size:	1048576
Level:	WARNING
Directory:	C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\Logs\
Proxy	
Server Name/IP:	
Bypass:	
User Name:	
Password:	

If you are using an **on-premises 'Azure DevOps Server' / TFS**, fill in the Provider Server fields as shown:

Protocol: Select http or https
Server Name/IP: Enter your server name or IP address
Port: Enter the Port number that DevOps/TFS is configured to use.
Base URL: Enter the final component of your DevOps/TFS url (everything after the server name/Port). E.g. tfs/DefaultCollection
User Name: Leave this field empty. Users will be prompted for their user name when using the integration within Enterprise Architect.
Password: Leave this field empty. Users will be prompted for a password when using the integration within Enterprise Architect. In this case the 'password' refers a personal access token (PAT). Personal access tokens are created from within DevOps via the 'Settings | Personal access tokens' option. Refer to the DevOps documentation for details.
If you prefer to set values in the 'User Name' and 'Password' fields, the values will be used in combination to access the data within DevOps; however, all current Enterprise Architect users of the DevOps installation will use the same set of credentials to read the external data.
Create Items / Modify Items / Post Discussions: Enable these checkboxes as necessary, to allow users of the DevOps Plug-in to create and/or update items and/or create Discussion posts in DevOps using Enterprise Architect.
Ignore SSL Errors: Leave this option unchecked.



Add Data Provider

Data Provider:

Enabled:	✓
Name: *	DevOps
Provider: *	Team Foundation Server
Prefix: *	tfs

Integration Plugin

Auto Start:	✓
Port: *	8081

Provider Server

Protocol: *	http
Server Name/IP: *	123.45.67.890
Port: *	8080
Base URL:	tfs/DefaultCollection
Maximum Request Time:	60 seconds
User Name:	sparxuser
Password:	●●●●●●●●●●●●●●●●
Create Items:	✓
Modify Items:	✓
Post Discussions:	✓
Ignore SSL Errors:	☐

Logging

File Count:	3
Max File Size:	1048576
Level:	WARNING
Directory:	C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\Logs\

Proxy

Server Name/IP:	
Bypass:	
User Name:	
Password:	

Note, the Protocol, Server, Port and Base URL fields are concatenated to make a web address; that is: *<protocol>://<server>:<port>/baseURL* (*baseURL* included if the field is not blank).

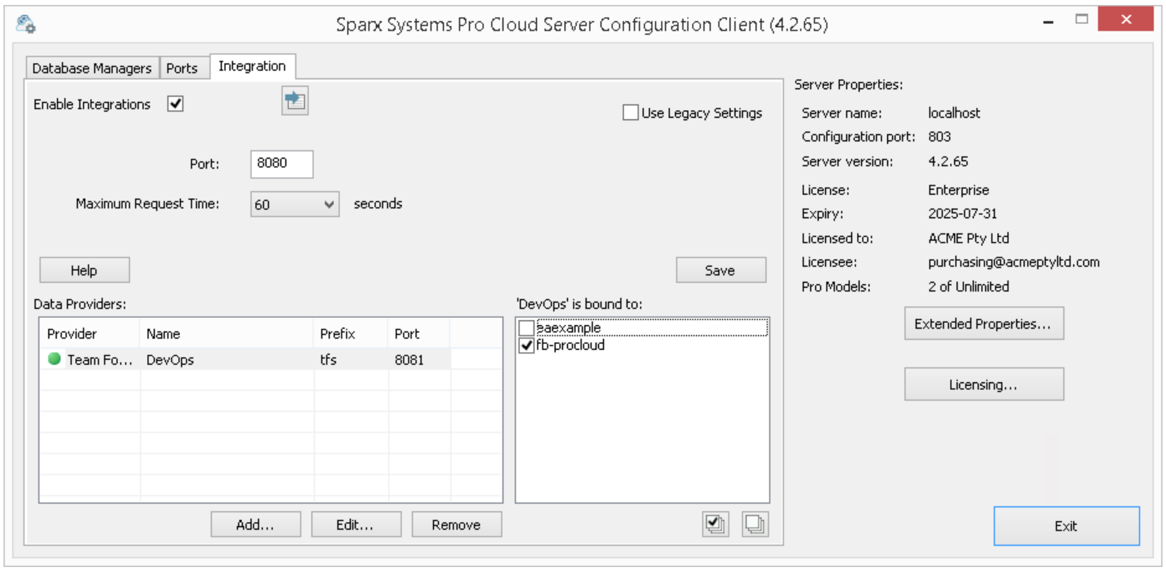
20. You can leave other fields blank or set to their default values. Click on the OK button to complete the configuration. This returns you to the 'Integration' tab of the Pro Cloud Server Configuration Client window.

Identify Enterprise Architect models to DevOps

On the 'Integration' tab of the Pro Cloud Server Configuration Client window, you will now see:

- On the lower left of the screen, the 'Data Providers' panel listing Team Foundation Server/DevOps as a data Provider
- On the lower right of the screen, the "DevOps' is bound to:' panel displaying a check list of the Enterprise Architect models from your Pro-enabled Database Manager(s)

You can either click on the checkbox against each Enterprise Architect model you want to bind to DevOps, or click on the ☑ to select all of them at once.


Sparx Systems Pro Cloud Server Configuration Client (4.2.65)

Database Managers | Ports | Integration

Enable Integrations [✓]

[ ] Use Legacy Settings

Port: 8080

Maximum Request Time: 60 seconds

Help | Save

Data Providers:

Provider	Name	Prefix	Port
● Team Fo...	DevOps	tfs	8081

Add... | Edit... | Remove

'DevOps' is bound to:

[ ] eaexample
[✓] fb-procloud

Server Properties:

Server name:	localhost
Configuration port:	803
Server version:	4.2.65
License:	Enterprise
Expiry:	2025-07-31
Licensed to:	ACME Pty Ltd
Licensee:	purchasing@acmeptyltd.com
Pro Models:	2 of Unlimited

Extended Properties... | Licensing... | Exit

Click on the Exit button.

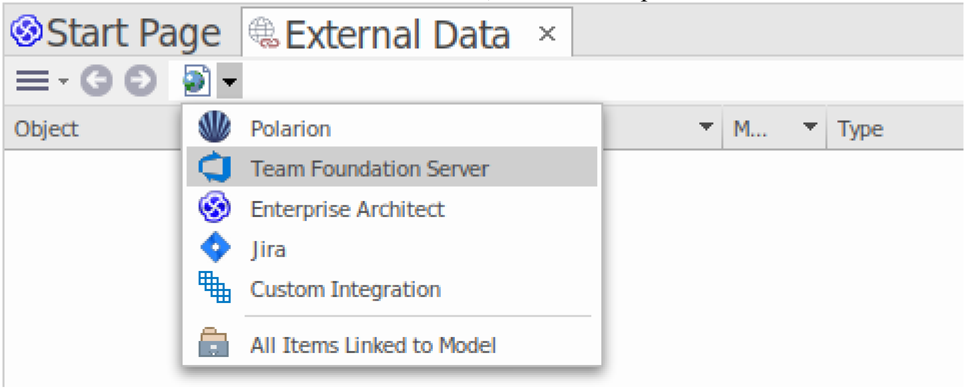
Restart the PCS Service to apply the changes you have made.

You can now go into one of your Enterprise Architect models and check the integration with DevOps.

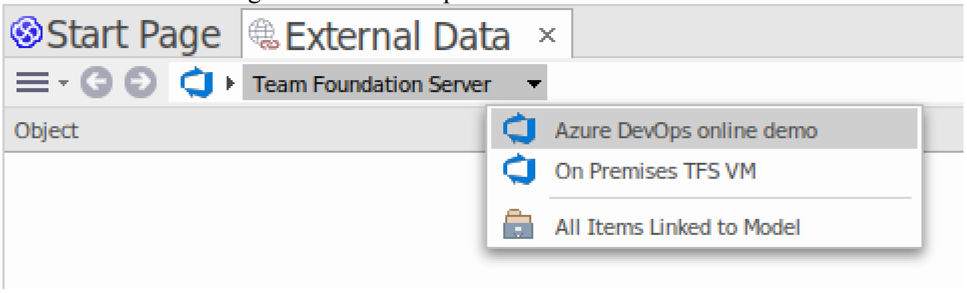
Test the Integration

In Enterprise Architect open one of the models you have identified as being bound to DevOps.

1. Select the ribbon option Specialize > Tools > System Integration > Open External Data.
2. In the toolbar of the External Data window, click on the drop-down arrow to the right of the 'globe' icon.
3. Select 'Team Foundation Server' from the list; this adds the provider to the breadcrumb trail in the toolbar.



4. Click on 'DevOps' in the breadcrumb trail to display a list of DevOps providers that have been configured and bound to this model. If nothing is listed then no provider has been bound to this model.



5. Click on one of the Projects which are defined in DevOps.
6. Select either 'Shared Queries' or 'My queries'

7. Select one of the defined queries to display all the items returned by that query in the list below.

(If you cannot see Team Foundation Server, the DevOps project/s or any defined queries, check through the procedures in this topic, or the broader and more detailed topics in the Integration Plug-ins section to see if there is a step you have missed or a prerequisite that is not satisfied. You can also check the Troubleshooting Help topic.)

Data Mapping

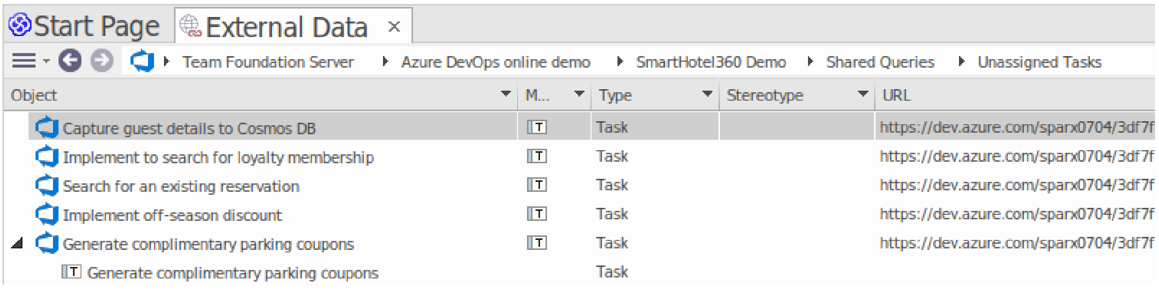
In order for DevOps and Enterprise Architect to smoothly exchange data items, you have to review and - where necessary - edit:

- What type of DevOps item corresponds to which type of Enterprise Architect element, and
- What property of a DevOps item corresponds to which property of an Enterprise Architect element.

These tasks are fully explained in the Configuration Help topic.

Review DevOps Data

When you tested the communication between Enterprise Architect and DevOps, on the External Data window, you opened the breadcrumb trail to show a list of DevOps projects, then selected either 'My Queries' or 'Shared Queries' and selected a query from the list. The work items are then listed in the body of the window.



Team Foundation Server ▸ Azure DevOps online demo ▸ SmartHotel360 Demo ▸ Shared Queries ▸ Unassigned Tasks

Object	M...	Type	Stereotype	URL
Capture guest details to Cosmos DB		Task		https://dev.azure.com/sparx0704/3df7f
Implement to search for loyalty membership		Task		https://dev.azure.com/sparx0704/3df7f
Search for an existing reservation		Task		https://dev.azure.com/sparx0704/3df7f
Implement off-season discount		Task		https://dev.azure.com/sparx0704/3df7f
Generate complimentary parking coupons		Task		https://dev.azure.com/sparx0704/3df7f
Generate complimentary parking coupons		Task		

If this list is too long, you can click on the 'Hamburger' icon in the window toolbar and select options such as:

- 'Linked Items' to show only those DevOps items that are linked to Enterprise Architect elements
- 'Items Linked to Current Context' to show only the DevOps items that are linked to the currently selected Enterprise Architect element or
- 'Unlinked items' to show only those DevOps items that have not yet been linked to Enterprise Architect elements

Select the appropriate option, if necessary, and then explore the content of selected DevOps items. Each of the Properties window, Notes window and Inspector window have separate tabs or versions of the window to display any properties, notes and features (respectively) of the selected item. These are illustrated in the External Item Details Help topic.

Working with DevOps items and Enterprise Architect elements

Having created a working communication between Enterprise Architect and DevOps, you can:

- Create a new element in Enterprise Architect linked to a DevOps item
- Create a new DevOps item linked to an element in Enterprise Architect
- Link an existing element in Enterprise Architect to a DevOps item
- Update the DevOps item with changes to the Enterprise Architect element
- Update the Enterprise Architect element with changes to the DevOps item
- Update all linked Enterprise Architect elements with any changes to the DevOps items

These actions are all explained in the Linking Items Help topic.

Walkthrough: Polarion Integration

Prerequisites

In order to set up integration between Polarion and Enterprise Architect, you must first:

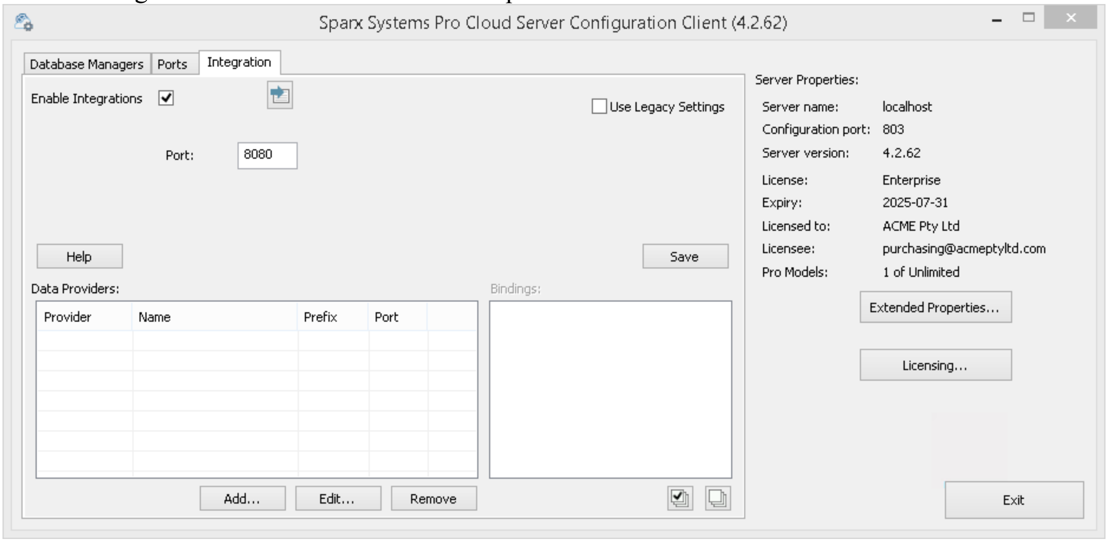
- Be using a licensed installation of Pro Cloud Server (Version 4.2.62 or later) with the Integration Plug-ins component
- Select the 'Enable Pro Features (OSLC, WebEA and Integration)' option in the Pro Cloud Server configuration for each Database Manager System hosting your Enterprise Architect models
- Have physical network access between the PCS server and the Polarion server
- Have user credentials to access the external data
- Be using Enterprise Architect Corporate, Unified or Ultimate edition v15.2 or later

We will assume that the Integration Server and the Polarion Integration Plug-in are to run on the same machine as the Pro Cloud Server

Identify Polarion to Pro Cloud Server and Enterprise Architect

Work through these steps:

8. Double-click on the 'Pro Cloud Config Client' Desktop icon to open the Pro Cloud Config Client window.
9. Select the 'Integration' tab and select the 'Enable Integrations' checkbox. In the 'Port' field type the number of the Port the Integration Server will listen on for requests from the Pro Cloud Server.



Sparx Systems Pro Cloud Server Configuration Client (4.2.62)

Enable Integrations ✓

Use Legacy Settings ☐

Port: 8080

Data Providers:

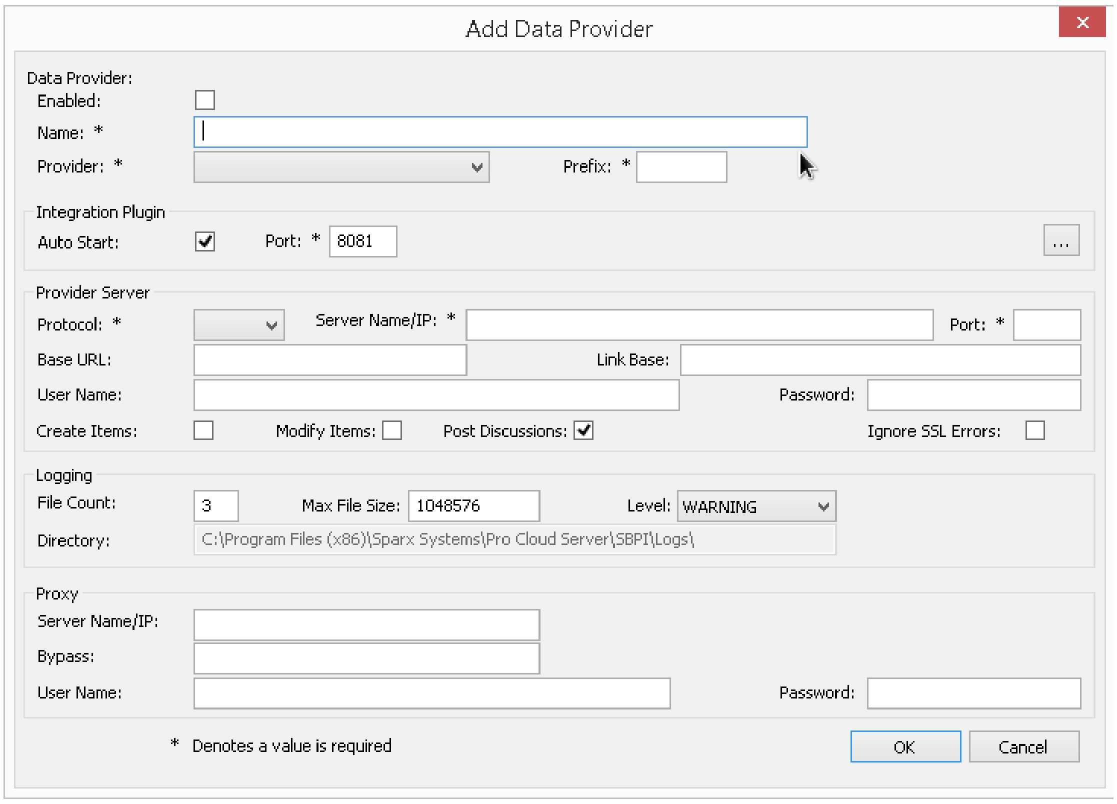
Provider	Name	Prefix	Port

Bindings:

Server Properties:

Server name:	localhost
Configuration port:	803
Server version:	4.2.62
License:	Enterprise
Expiry:	2025-07-31
Licensed to:	ACME Pty Ltd
Licensee:	purchasing@acmeptyltd.com
Pro Models:	1 of Unlimited

10. Click on the Add button. The 'Add Data Provider' dialog displays.


Add Data Provider

Data Provider:

Enabled:	☐
Name: *	
Provider: *	
Prefix: *	

Integration Plugin

Auto Start:	✓
Port: *	8081

Provider Server

Protocol: *	
Server Name/IP: *	
Port: *	
Base URL:	
Link Base:	
User Name:	
Password:	
Create Items:	☐
Modify Items:	☐
Post Discussions:	✓
Ignore SSL Errors:	☐

Logging

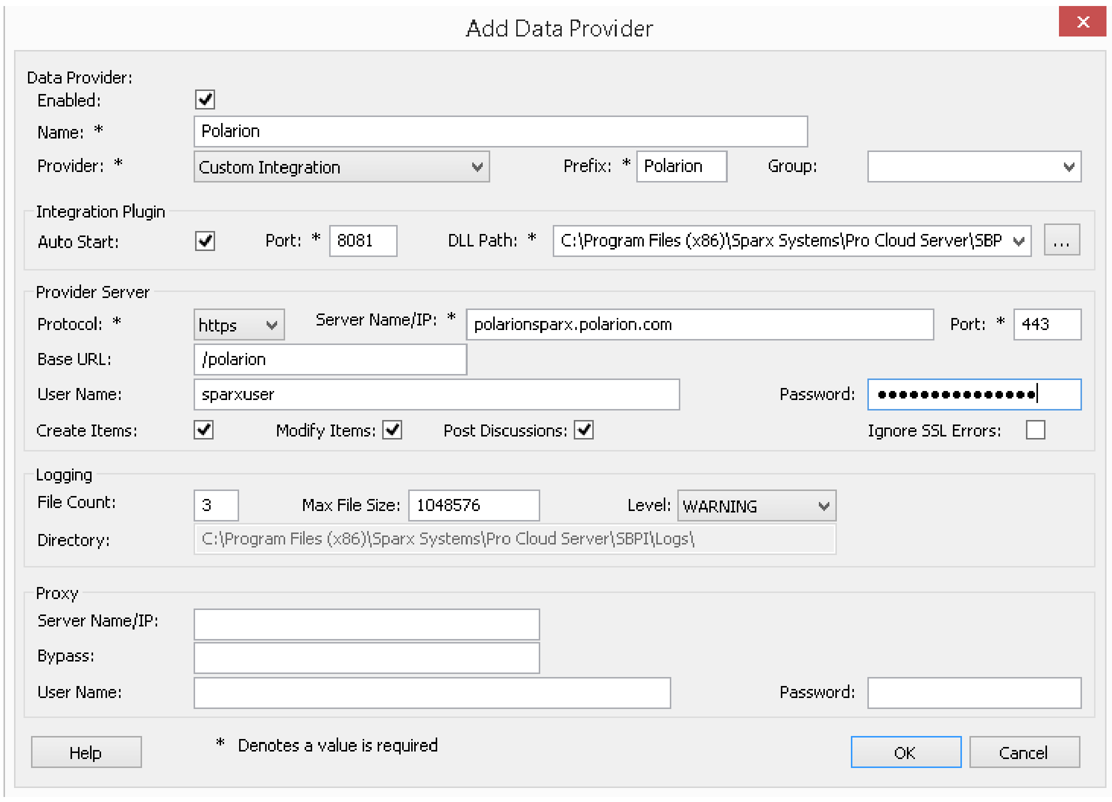
File Count:	3
Max File Size:	1048576
Level:	WARNING
Directory:	C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\Logs\

Proxy

Server Name/IP:	
Bypass:	
User Name:	
Password:	

* Denotes a value is required

11. Select the 'Enabled' checkbox.
12. In the 'Name' field, type an appropriate connection name, such as 'Polarion'.
13. In the 'Provider' field, click on the drop-down arrow and select 'Custom Integration'.
14. In the 'Prefix' field, enter an appropriate prefix, such as 'Polarion'.
15. In the 'Integration Plugin' panel, in the 'Port' field, type the number of the Port that the Integration Server will use to communicate with the Polarion Plug-in.
16. Click on the ellipsis button next to the 'DLL Path' field. Browse and select the 'PolarionSbpi.dll' file. This is located within the Pro Cloud Server Installation folder. E.g. C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\PolarionSbpi.dll
17. The fields in steps 10 to 13 make up the components of your Polarion web address; that is: *<protocol>://<server>:<port>/baseURL*
    In the 'Protocol' field, click on the drop-down arrow and select the protocol that the Polarion Plug-in will use to communicate with Polarion.
18. In the 'Server Name/IP' field, type the server name or IP address that the Polarion Plug-in will use to communicate with Polarion.
19. In the 'Port' field type the number of the Port that the Polarion Plug-in will use to communicate with Polarion.
20. In the 'Base URL' field enter the final portion of your Polarion URL (anything after the server name and port)
21. If you leave the 'User' and 'Password' fields blank, then each Enterprise Architect user will be prompted for their personal Polarion credentials, which can give a better usability.
22. If you prefer to set values in the 'User Name' and 'Password' fields, the values will be used in combination to access the data within Polarion. All current Enterprise Architect users of the Polarion installation will use the same set of credentials to read the external data.
23. Select the 'Create Items', 'Modify Items' and/or 'Post Discussions' checkboxes as necessary, to allow users of the Polarion Plug-in to create and/or update items and/or create Discussion posts in Polarion using Enterprise Architect.
24. If you want the Polarion Plug-in to ignore SSL-related errors that occur as a result of communicating with Polarion (e.g. if you are using a self-signed SSL certificate), select the 'Ignore SSL Errors' checkbox.


Add Data Provider

Data Provider:

Enabled:	✓
Name: *	Polarion
Provider: *	Custom Integration
Prefix: *	Polarion
Group:	

Integration Plugin

Auto Start:	✓
Port: *	8081
DLL Path: *	C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBP

Provider Server

Protocol: *	https
Server Name/IP: *	polarionsparx.polarion.com
Port: *	443
Base URL:	/polarion
User Name:	sparxuser
Password:	•••••••••••••••
Create Items:	✓
Modify Items:	✓
Post Discussions:	✓
Ignore SSL Errors:	

Logging

File Count:	3
Max File Size:	1048576
Level:	WARNING
Directory:	C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\Logs\

Proxy

Server Name/IP:	
Bypass:	
User Name:	
Password:	

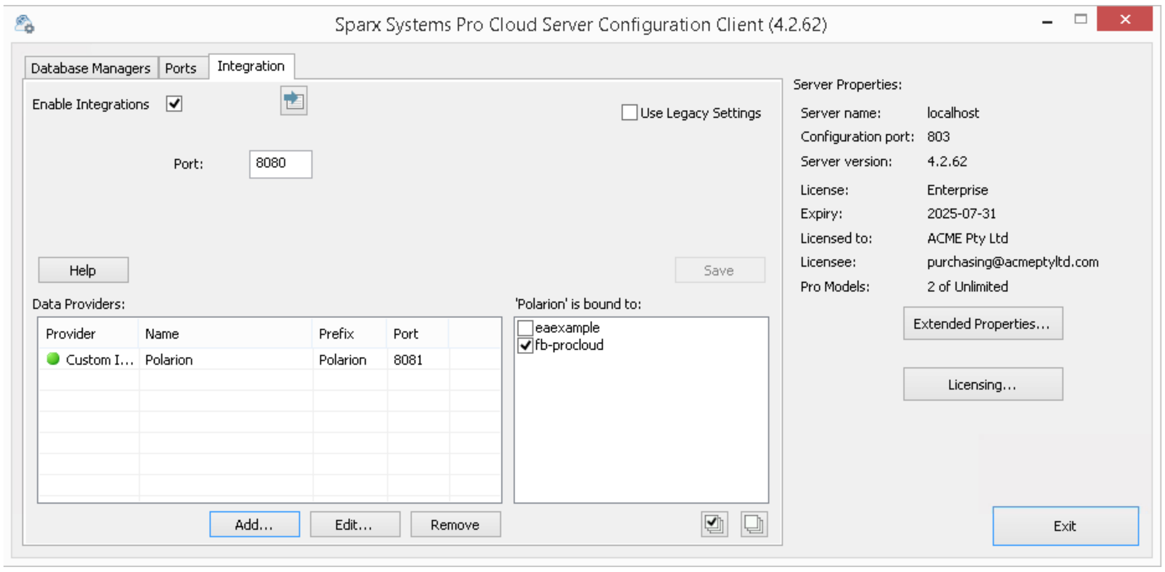
25. You can leave other fields blank or set to their default values. Click on the OK button to complete the configuration. This returns you to the 'Integration' tab of the Pro Cloud Server Configuration Client window.

Identify Enterprise Architect models to Polarion

On the 'Integration' tab of the Pro Cloud Server Configuration Client window, you will now see:

- On the lower left of the screen, the 'Data Providers' panel listing the Custom Integration for Polarion as a data Provider
- On the lower right of the screen, the ''Polarion' is bound to:' panel displaying a check list of the Enterprise Architect models from your Pro-enabled Database Manager(s)

You can either click on the checkbox against each Enterprise Architect model you want to bind to Polarion, or click on the ☑ to select all of them at once.



Sparx Systems Pro Cloud Server Configuration Client (4.2.62)

Database Managers | Ports | Integration

Enable Integrations ✓

Use Legacy Settings ☐

Port: 8080

Help | Save

Data Providers:

Provider	Name	Prefix	Port
● Custom I...	Polarion	Polarion	8081

Add... | Edit... | Remove

'Polarion' is bound to:

☐ eaexample
✓ fb-procloud

Server Properties:

Server name:	localhost
Configuration port:	803
Server version:	4.2.62
License:	Enterprise
Expiry:	2025-07-31
Licensed to:	ACME Pty Ltd
Licensee:	purchasing@acmeptyltd.com
Pro Models:	2 of Unlimited

Extended Properties... | Licensing... | Exit

Click on the Exit button.

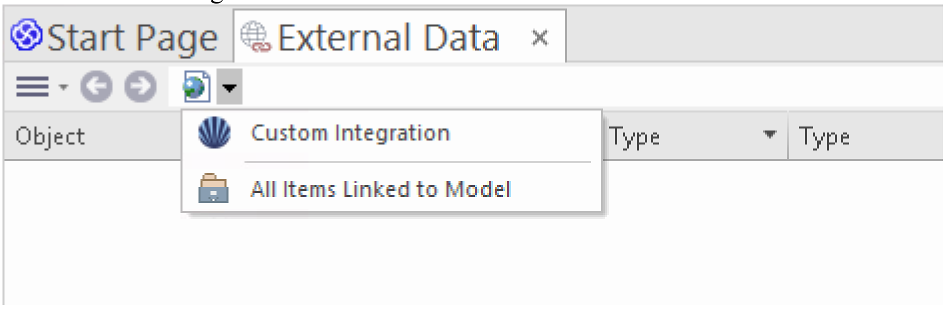
Restart the PCS Service to apply the changes you have made.

You can now go into one of your Enterprise Architect models and check the integration with Polarion.

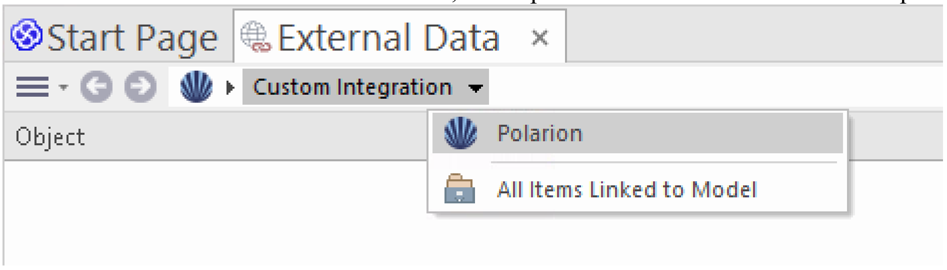
Test the Integration

In Enterprise Architect open one of the models you have identified as being bound to Polarion.

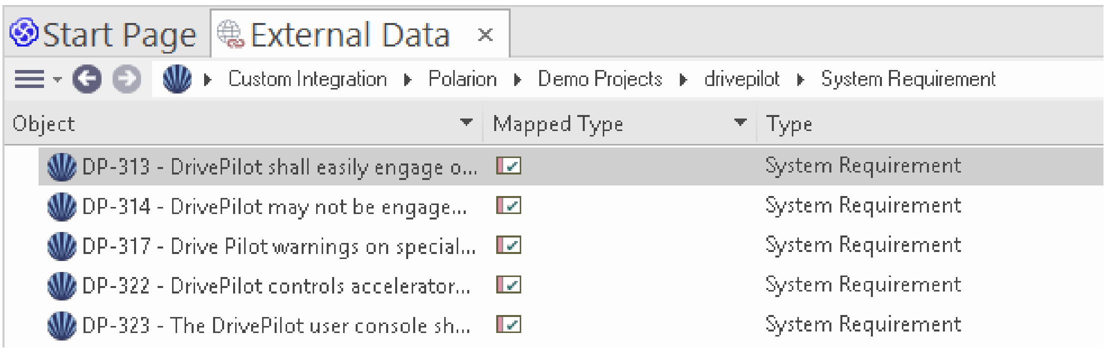
26. Select the ribbon option Specialize > Tools > System Integration > Open External Data.
27. In the toolbar of the External Data window, click on the drop-down arrow to the right of the 'globe' icon.
28. Select 'Custom Integration' from the list.



29. Click on 'Polarion' in the breadcrumb trail, this represents the root of the Polarion repository.



30. Continue using the breadcrumb options to drill down into the Polarion projects and access Work Items (Tasks, Issues, etc).


Start Page | External Data

Custom Integration ▸ Polarion ▸ Demo Projects ▸ drivepilot ▸ System Requirement

Object	Mapped Type	Type
DP-313 - DrivePilot shall easily engage o...	✓	System Requirement
DP-314 - DrivePilot may not be engage...	✓	System Requirement
DP-317 - Drive Pilot warnings on special...	✓	System Requirement
DP-322 - DrivePilot controls accelerator...	✓	System Requirement
DP-323 - The DrivePilot user console sh...	✓	System Requirement

Data Mapping

In order for Polarion and Enterprise Architect to smoothly exchange data items, you have to review and - where necessary - edit:

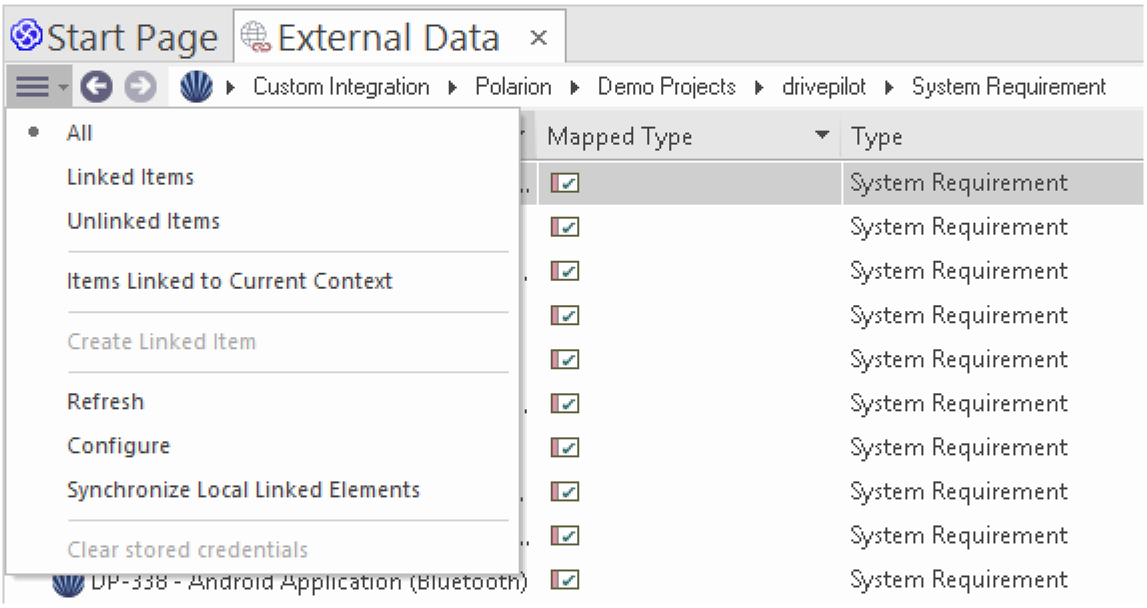
- What type of Polarion item corresponds to which type of Enterprise Architect element, and
- What property of a Polarion item corresponds to which property of an Enterprise Architect element.

These tasks are fully explained in the Configuration Help topic.

Review Polarion Data

When you tested the communication between Enterprise Architect and Polarion, on the External Data window, you used the breadcrumb trail to view lists of work items (e.g. Tasks).

If this list is too long, you can click on the 'Hamburger' icon in the window toolbar.

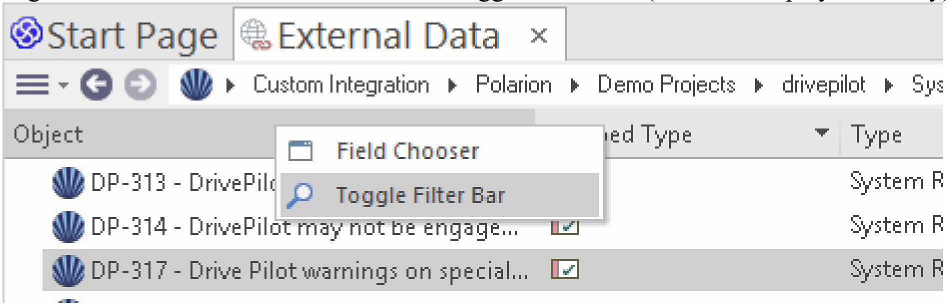

Start Page
External Data
Custom Integration ▸ Polarion ▸ Demo Projects ▸ drivepilot ▸ System Requirement
All
Linked Items
Unlinked Items
Items Linked to Current Context
Create Linked Item
Refresh
Configure
Synchronize Local Linked Elements
Clear stored credentials
Mapped Type
Type
System Requirement
DP-338 - Android Application (Bluetooth)

This menu provides options such as:

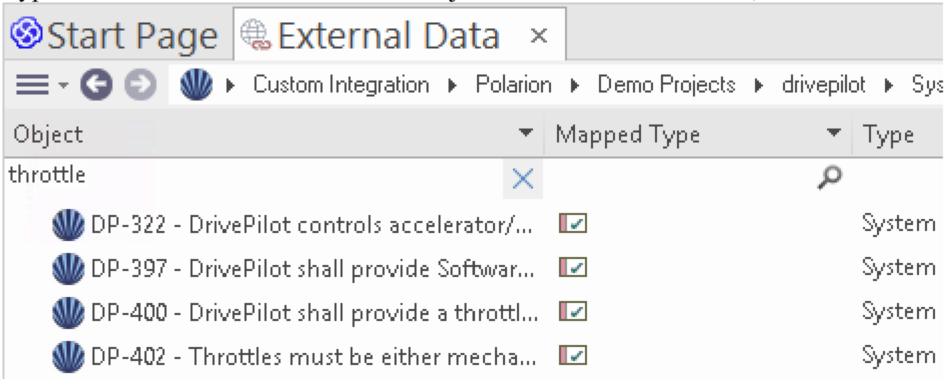
- 'Linked Items' to show only those Polarion items that are linked to Enterprise Architect elements
- 'Unlinked items' to show only those Polarion items that have not yet been linked to Enterprise Architect elements, or
- 'Items Linked to Current Context' to show only the Polarion items that are linked to the currently selected Enterprise Architect element

It's also possible to apply text based filters to the External Data table contents.

1. Right click the column header and select 'Toggle Filter Bar' (if it's not displayed already).



2. Type some text into the filter/search field just below the column name,



Object	Mapped Type	Type
throttle		
DP-322 - DrivePilot controls accelerator/...	✓	System
DP-397 - DrivePilot shall provide Softwar...	✓	System
DP-400 - DrivePilot shall provide a throttl...	✓	System
DP-402 - Throttles must be either mecha...	✓	System

When selecting an item from the list each of the Properties window, Notes window, Inspector window and 'Discuss' tab of the Discuss & Review window have separate tabs or versions of the window to display any properties, notes, features and discussions (respectively) of the selected item. These are illustrated in the External Item Details Help topic.

Working with Polarion items and Enterprise Architect elements

Having created a working communication between Enterprise Architect and Polarion, you can:

- Create a new element in Enterprise Architect linked to a Polarion item
- Create a new Polarion item linked to an element in Enterprise Architect
- Link an existing element in Enterprise Architect to a Polarion item
- Update the Polarion item with changes to the Enterprise Architect element
- Update the Enterprise Architect element with changes to the Polarion item
- Update all linked Enterprise Architect elements with any changes to the Polarion items

These actions are all explained in the Linking Items Help topic.

Note, when creating a Polarion item from Enterprise Architect, it is placed in the Polarion Work Items group (as a new job to be done).

Troubleshooting

Integration Server messages

There are certain error messages that could display during the definition of the Integration Server settings; most are self explanatory. This table describes the most common error messages.

Error Message	Description
Please enter a value for [field_names]. The Integration Server can only be saved (while enabled) when all mandatory fields have a value	Reason: When the 'Enabled' flag is checked, all remaining Integration Server fields are mandatory. This message occurs when one or more Integration Server fields are empty and the 'Enabled' flag has been checked. [field_names] represents a placeholder for a comma separated list of field names that are empty.
The executable name of SBPI.EXE was expected	Reason: The 'Executable Path' field has a value but it does not include the filename "...\SBPI.exe", which is the only valid filename that can be used.
The specified Integration Server path of [full_path] does not exist or is invalid	Reason: The 'Executable Path' field has a value but the path portion of the specified value is not valid or does not exist on the Pro Cloud Server machine.
The specified Port is a duplicate of [duplicate_plugin_name]	Reason: The 'Port' field has been assigned a Port number, but it is the same as a Port being used by another Integration Data Provider identified by the name [duplicate_plugin_name].

Integration Data Provider messages

There are certain error messages that could display during the definition of an Integration Data Provider; most are self explanatory. This table describes the most common error messages.

Error Message	Description
Please enter a value for [field_names]. An Integration Provider cannot be saved without all mandatory fields being assigned a value.	Reason: To minimize the chance of defining an invalid Data Provider, the application ensures that all mandatory fields are given a value. [field_names] represents a placeholder for a comma separated list of field names that are empty.
Ports need to be unique across all Providers, the port value of [new_port] has already been used by [other_provider]	Reason: A Port number has been defined for the current Data Provider; however, the value is a duplicate of another provider's Port.
The specified execution path for the provider does not exist or is invalid, [full_path]	Reason: The 'Execution Path' for the data provider has been assigned a value, but either the path or filename, or both, cannot be found on the local machine.

The specified Port is a duplicate of the Integration Server	Reason: A Port number has been defined for the current Data Provider, but the value is a duplicate of the Port used by the Integration Server.
Failed to rename config file [old_filename] to [new_filename]	Reason: When the user changes the config filename of an existing Data Provider, the Cloud Server must rename the old filename to the new one. This message will be shown if the rename task is not successful (which can occur if the file is in use by another process) and in this case manual intervention is the only way to resolve this issue.
Prefixes need to be unique across all Providers, the prefix value of [new_prefix] has already been used by [other_provider]	Reason: A Prefix has been defined for the current Data Provider, but the value is a duplicate of the prefix used by another provider.
Config filenames need to be unique across all Providers, the config filename of [new_filename] has already been used by [other_provider]	Reason: A Config filename has been defined for the current Data Provider, but the value is a duplicate of the config filename used by another provider.

General Troubleshooting

This table provides general advice to help identify and resolve common issues with the Integration component of the Pro Cloud Server.

Problem Description
Problem: After making changes to the Integration definitions of the Server or Data Providers, the Enterprise Architect users do not notice any differences. Solution: After changing the Integration configuration, was the Pro Cloud Server restarted? If not, restart the Pro Cloud Server.
Problem: A newly configured Integration Data Provider is not listed in the navigation breadcrumbs of Enterprise Architect (after the Pro Cloud Server has been restarted). Solution: Confirm that the currently open model has a Binding entry for the Data Provider. For details, see the 'Bindings' field in the *Data Providers* table of the Integration Plug-ins Help topic.
Problem: Selecting items within the Integration navigation breadcrumb of Enterprise Architect seem to have no effect. Solution: It is possible that errors (or warnings) are occurring that are logged to the System Output window, therefore make sure the System Output window is displayed while using the External Data window.
Problem: General errors seem to be occurring; however, no useful information is shown in the System Output window. Solution: Each Integration Data Provider has the capability to write detailed entries to a log file; to ensure that the most detailed information is written, change the log level for the Data Provider to SYSTEM (restart the Pro Cloud Server) and then retry the same action in Enterprise Architect. Then review the Data Provider's log file; that is, in C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI\Logs, on the Pro Cloud Server.

Problem: Generic or blank errors received from the Plug-in. Solution: Check that there are no other applications of services using the same Port as the SBPI.exe as defined in the main Integration configuration tab. If another program is using the same Port then the error messages will come from that application and not Pro Cloud Server or SBPI. A list of open Ports and the applications using them can be found in Windows 'Resource Monitor'.

Information Accessed

Each of the Integration Plug-ins returns information based on a 'Filter' or position within the external product's data. Some products, such as Enterprise Architect, Jira and DevOps, provide a mechanism to customize the data returned, whilst others simply return all information at a particular position within the external provider's application.

Information Accessed from each Provider

Provider	Information returned
Application Lifecycle Manager	Information returned based on the internal list for Defects, Requirements and Tests.
AutoDesk	Information returned based on the contents of Hubs | Projects | Folders.
Bugzilla	Information returned based on the contents of Product | Component | <all items in component>.
Dropbox	Information returned based on the contents of Folders.
Enterprise Architect	When you connect into an external Enterprise Architect repository via an Integration, in the 'External Data' bread-crumb, you will be offered a menu with 'Browse' or 'Search'. On selecting 'Search' it returns a list of the searches defined in the local repository. On selecting a specific search, the Item List will display the results from the external repository's data.
Jazz	Information returned based on the contents of (DoorsNG) - Folders.
Jira	Presents a list of 'Favorite Filters'. See the menu option 'Issues | Manage Filters'.
Salesforce	Presents all item types that have a 'List View'. In a default install these include: Accounts, Assets, Campaigns, Cases, Groups, Contacts, Contracts, Leads, and Opportunities.
ServiceNow	Presents a list of user-defined filters, grouped by the table they are based on.
DevOps / Team Foundation Server	Presents a list of DevOps / TFS global queries and 'My ...' queries.
Wrike	Information returned based on the contents of Accounts | Folders.

Notes

For Enterprise Architect, Jira or DevOps the filters must be configured prior to Enterprise Architect requesting that an Integration Plug-in should connect to them.

Navigate External Data

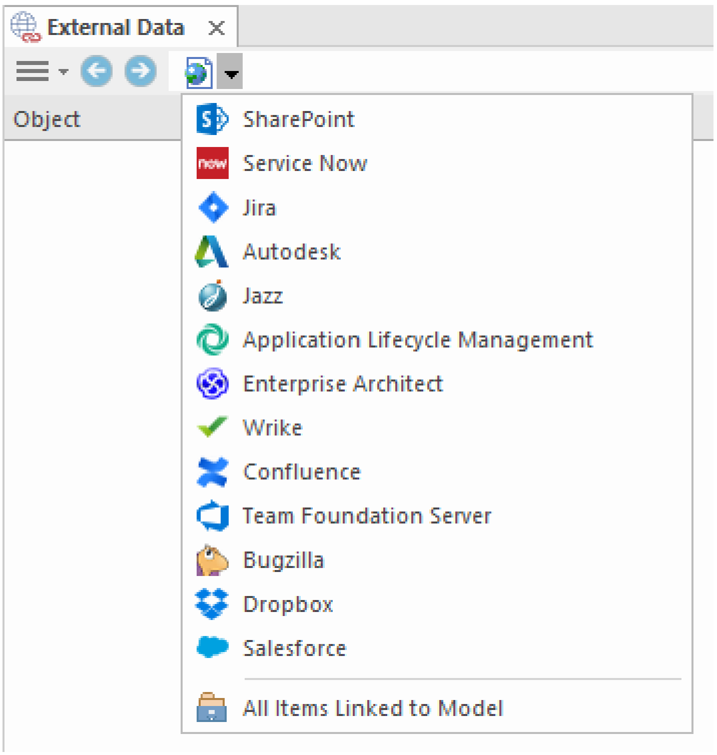
Each Integration Provider can be navigated to show lists of external items that can be linked to Enterprise Architect. Each provider might provide a slightly different mechanism for navigation, based on how it stores its data. Some provide a simple folder hierarchy, whilst others provide for user-defined filters. See the *What data is returned by Integration Plug-ins* table in the Install and Configure Help topic for details of each provider.

Access

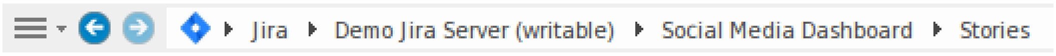
Ribbon	Specialize > Tools > System Integration > Open External Data

Navigate the Hierarchy

Begin navigation by selecting the provider type from the right-hand drop-down menu.

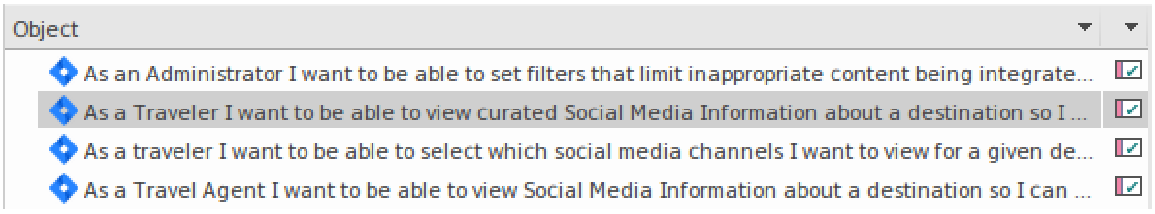


Next, each provider type offers a slightly different navigation system; for example, Dropbox allows for browsing the folder structure, whilst Jira gives a list of projects followed by a list of user queries.



Item List

At each navigation level, if available, a list of items corresponding to the navigation level will populate the left-hand panel.


Object

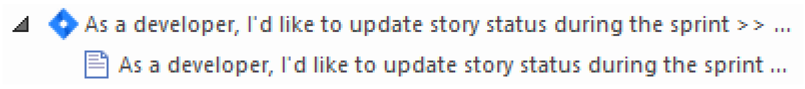
As an Administrator I want to be able to set filters that limit inappropriate content being integrate...

As a Traveler I want to be able to view curated Social Media Information about a destination so I ...

As a traveler I want to be able to select which social media channels I want to view for a given de...

As a Travel Agent I want to be able to view Social Media Information about a destination so I can ...

Any local Enterprise Architect elements that are linked to the external item will be shown as a child of the external item.


As a developer, I'd like to update story status during the sprint >> ...

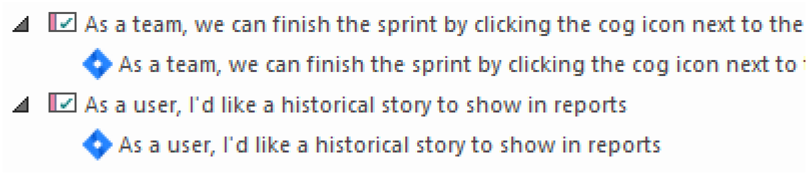
As a developer, I'd like to update story status during the sprint ...

All Items Linked to Model

At each navigation level there is an option to show all local Enterprise Architect elements that are linked to the selected external data source. Select 'All Items Linked to Model' and choose the number of days prior to today (7, 30 or 90 days, or 'All') from which to collect the information.

Choose 'All Items Linked to Model' on the root navigation level to see the linked items for all external providers.

This view differs from the regular list as it shows the local Enterprise Architect element on top, and the external linked item as a child of the local element. All the same context menu items are available in the view.


As a team, we can finish the sprint by clicking the cog icon next to the

As a team, we can finish the sprint by clicking the cog icon next to

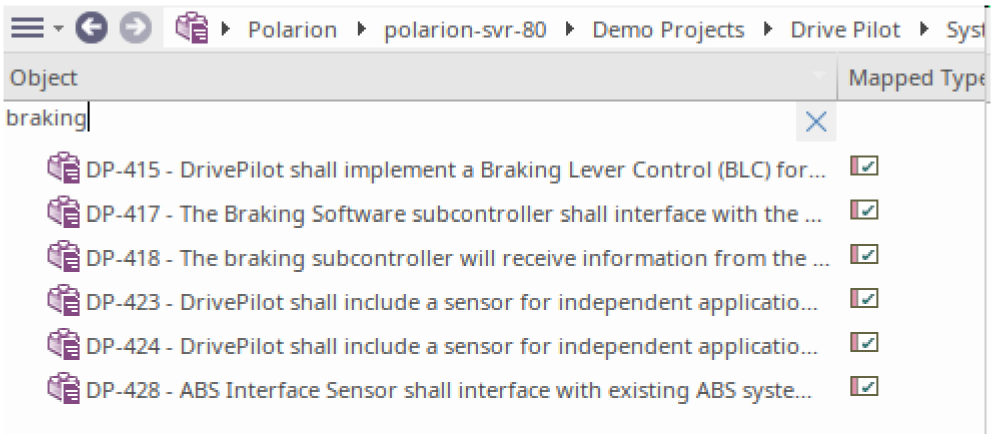
As a user, I'd like a historical story to show in reports

As a user, I'd like a historical story to show in reports

Filter the List

The list of external items can be filtered using the Filter Bar. To activate the Filter Bar, right-click on the header of the list and select 'Toggle Filter Bar'.

Columns can be filtered by typing text into the Filter Bar. The list will be filtered to show only those items with text containing the filter text.


Polarion ▸ polarion-svr-80 ▸ Demo Projects ▸ Drive Pilot ▸ Syst

Object	Mapped Type
braking	
DP-415 - DrivePilot shall implement a Braking Lever Control (BLC) for...	✓
DP-417 - The Braking Software subcontroller shall interface with the ...	✓
DP-418 - The braking subcontroller will receive information from the ...	✓
DP-423 - DrivePilot shall include a sensor for independent applicatio...	✓
DP-424 - DrivePilot shall include a sensor for independent applicatio...	✓
DP-428 - ABS Interface Sensor shall interface with existing ABS syste...	✓

Show Items Linked to Current Context

In the integrations menu, select 'Items Linked to Current Context' to show only the external items that are linked to the currently selected local element. That is, select an item in the Browser window and see the external items that are linked to it.

This view is the same as for 'All Items Linked to Model'.

External Item Details

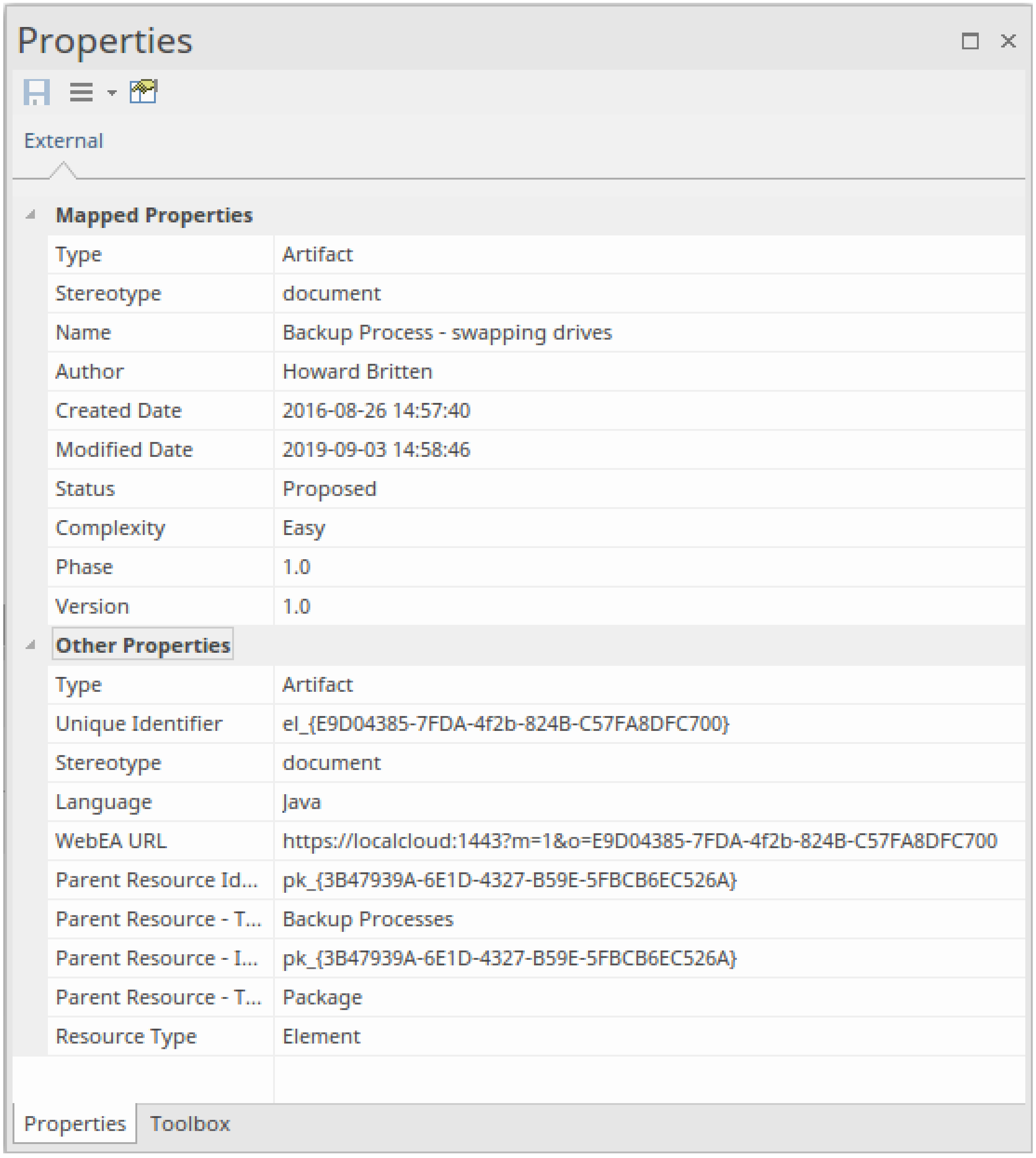
When an external item is selected in the External Data window, the item's details are retrieved and displayed in an 'External' tab of the Inspector window, and in External versions of the appropriate Properties, Notes and Discuss & Review windows.

Inspector Window

The 'External' tab of the Inspector window displays only when you are reviewing external items. It behaves in much the same way as the 'Details' tab of the Inspector window, revealing the external element's relationships, features, requirements, Tagged Values, project maintenance items and related files. However, the 'External' tab shows only categories for which items exist, and does not list all the possible but empty categories.

Properties

Click on an external item in the list to view its properties in the 'External' version of the Properties window. This window shows all available properties of the external item, as defined by the external provider.

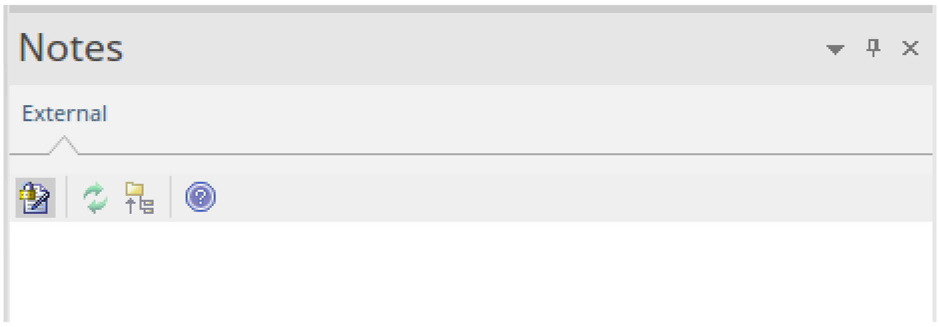


Mapped Properties	
Type	Artifact
Stereotype	document
Name	Backup Process - swapping drives
Author	Howard Britten
Created Date	2016-08-26 14:57:40
Modified Date	2019-09-03 14:58:46
Status	Proposed
Complexity	Easy
Phase	1.0
Version	1.0
Other Properties	
Type	Artifact
Unique Identifier	el_{E9D04385-7FDA-4f2b-824B-C57FA8DFC700}
Stereotype	document
Language	Java
WebEA URL	https://localcloud:1443?m=1&o=E9D04385-7FDA-4f2b-824B-C57FA8DFC700
Parent Resource Id...	pk_{3B47939A-6E1D-4327-B59E-5FBCB6EC526A}
Parent Resource - T...	Backup Processes
Parent Resource - I...	pk_{3B47939A-6E1D-4327-B59E-5FBCB6EC526A}
Parent Resource - T...	Package
Resource Type	Element

Properties that will be used when creating a linked local element are shown in the top group 'Mapped Properties', while all other properties are grouped in 'Other Properties'. The property mapping can be configured, as described in the *Configuration* Help topic.

Notes

The external item's 'Notes' or 'Description' text is shown in the 'External' version of the Notes window.



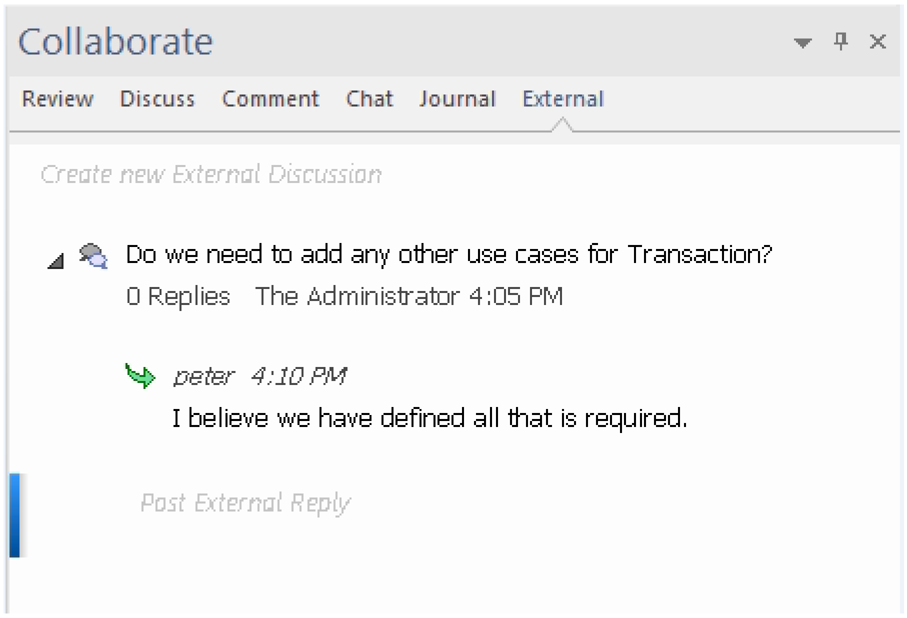
Note that not all external providers have 'Notes' or 'Description' fields.

This version of the Notes window has a different toolbar, containing icons to:

- Toggle the editing lock on the notes
- Synchronize the notes in the Enterprise Architect view and the external source
- Push the edited note text to the external source
- Display the online Enterprise Architect User Guide

Discussion

Select an external item from the list to view and participate in its discussions in the 'External' tab of the Discuss & Review window. The 'External' tab resembles the 'Discuss' tab and behaves in the same way.



Do we need to add any other use cases for Transaction?
0 Replies The Administrator 4:05 PM

peter 4:10 PM
I believe we have defined all that is required.

New discussions can be posted by double-clicking on the *Create new External Discussion* text, or by right-clicking in the tab and selecting the 'Create New External Discussion' menu option.

Note:

- To post an external discussion the Integration Provider must be configured to allow this on the Pro Cloud Server (allowed by default)
- Not all external providers have discussions or comments

Configuration

Each Integration Provider comes with a set of default mapping values that determines firstly what type of local element is created in Enterprise Architect, and secondly which fields are copied to the new element. These mappings are configurable for each client model.

Permission

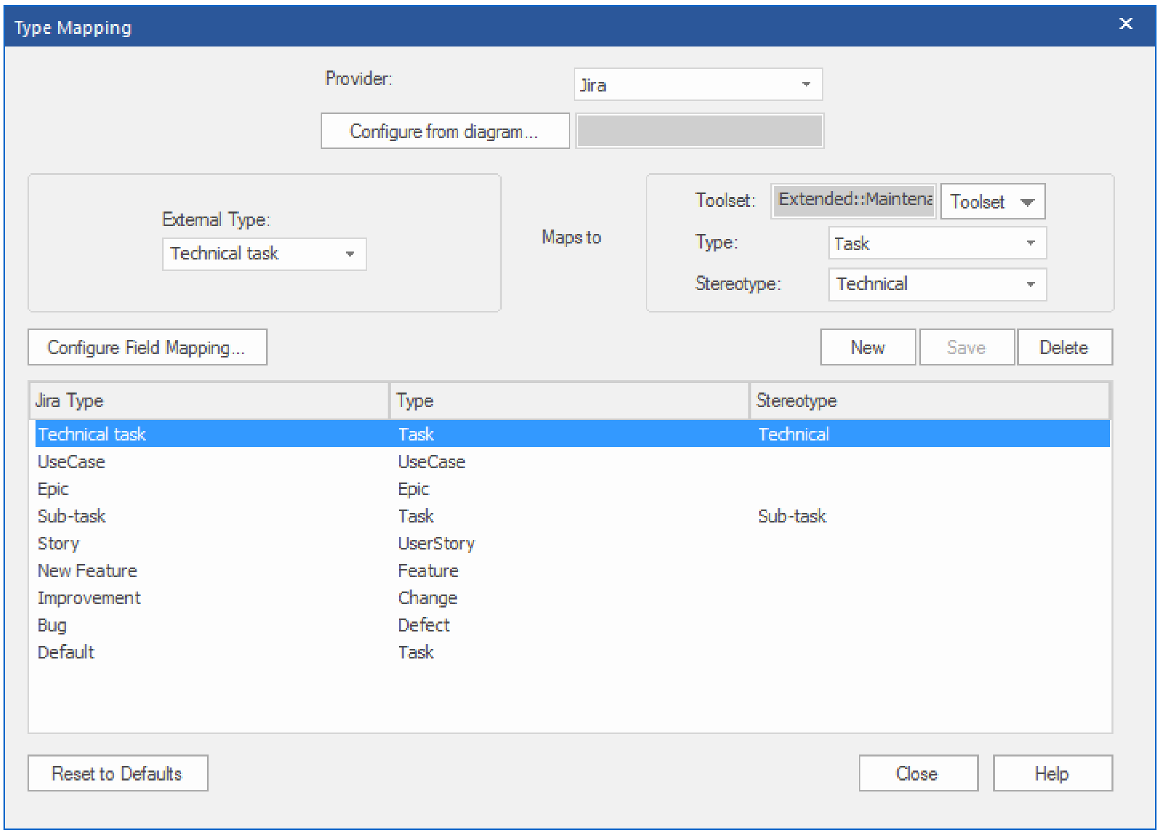
You must have 'Configure External Data Sources' permission to access this functionality.

Access

External Data window toolbar	'Hamburger' Menu > Configure

Type Mapping

The 'Type Mapping' dialog defines the element type that will be created when users select the 'Create Local Element' option for an external element.



Provider: Jira

External Type: Technical task

Toolset: Extended::Maintena

Type: Task

Stereotype: Technical

Jira Type	Type	Stereotype
Technical task	Task	Technical
UseCase	UseCase	
Epic	Epic	
Sub-task	Task	Sub-task
Story	UserStory	
New Feature	Feature	
Improvement	Change	
Bug	Defect	
Default	Task	

To create a new mapping:

1. Click on the New button, then click on the 'External Type' drop-down arrow and select the type of the element to

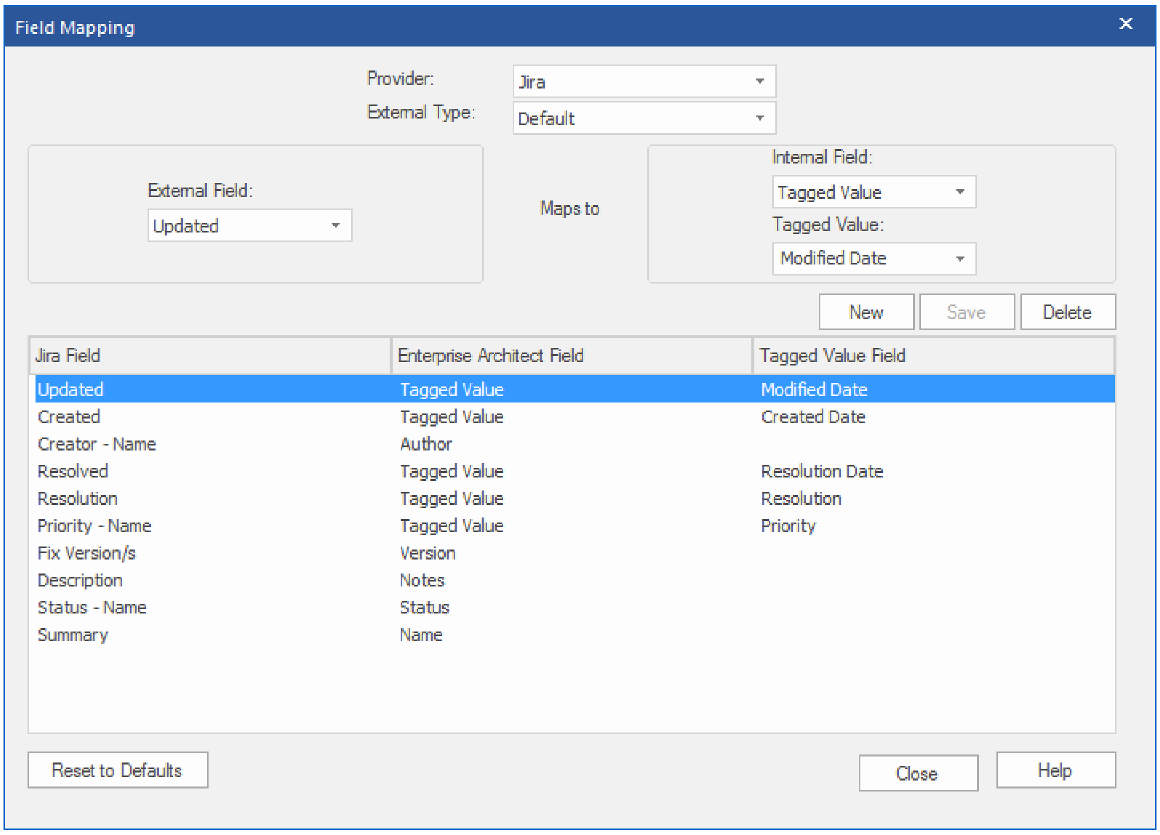
map from the external source.

2. Click on the 'Toolset' drop-down arrow and select the appropriate Enterprise Architect Toolset, then click on the 'Type' drop-down arrow and select the Enterprise Architect element type that the external element type maps to.
3. If appropriate, also click on the 'Stereotype' drop-down arrow and select the stereotype for the extended Enterprise Architect element type that the external element type maps to.
4. Click on the Save button.

Note that if the external element type has been linked as an <<ExternalReference>> element, any Type Mapping settings are ignored; see the Linking Items Help topic.

Field Mapping

The 'Field Mapping' dialog defines the element fields that will be updated for a mapped element when users select the 'Create Local Element' option. You display this dialog by clicking on the Configure Field Mapping button on the 'Type Mapping' dialog, having mapped an external element type to an Enterprise Architect element (or selected an existing mapped element type).



Jira Field	Enterprise Architect Field	Tagged Value Field
Updated	Tagged Value	Modified Date
Created	Tagged Value	Created Date
Creator - Name	Author	
Resolved	Tagged Value	Resolution Date
Resolution	Tagged Value	Resolution
Priority - Name	Tagged Value	Priority
Fix Version/s	Version	
Description	Notes	
Status - Name	Status	
Summary	Name	

The process generally maps the fields for a specific type of element; however, if you want to map certain fields for all types of element, click on the 'External Type' drop-down arrow on the 'Field Mapping' dialog, and select the value 'Default'.

To map the fields:

5. Click on the New button, then click on the 'External Field' drop-down arrow and select the name of the field to map from the external source.
6. Click on the 'Internal Field' drop-down arrow and select the appropriate Enterprise Architect field name that the external element field maps to.
7. If the internal field is a Tagged Value field, also click on the 'Tagged Value' drop-down arrow and select the Tagged Value that the external element field maps to.
8. Click on the Save button.

Notes

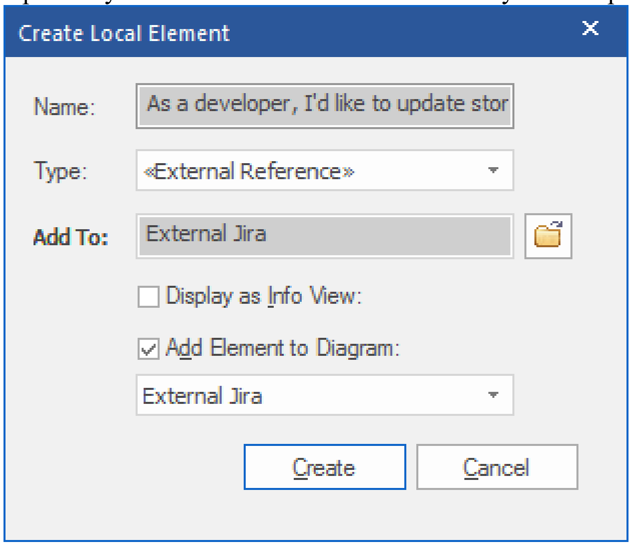
- Each mapping can be reset to default values by clicking on the Reset to Defaults button
- If no type mapping is defined for a given element type, then the mapping for type 'Default' will be used

Linking Items

Create Linked Element in Enterprise Architect

To create a new element in the Enterprise Architect model linked to the external item:

9. Right-click on an external item in the list in the External Data window.
10. Select 'Create Local Element'.
11. Select either the default type or «ExternalReference» (which ignores any Type Mapping; see the Configuration Help topic).
12. Select which Package the element will be created in.
13. Optionally add the new element to the currently-active open diagram.


Create Local Element

Name: As a developer, I'd like to update stor

Type: «External Reference»

Add To: External Jira

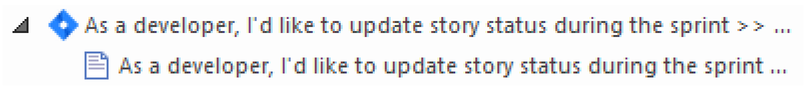
[ ] Display as Info View:

[✓] Add Element to Diagram:

External Jira

The local element properties will be created from the values defined in the 'Field Mapping' dialog, described in the Configuration Help topic.

Once linked, the local element will show under the external item:


As a developer, I'd like to update story status during the sprint >> ...

As a developer, I'd like to update story status during the sprint ...

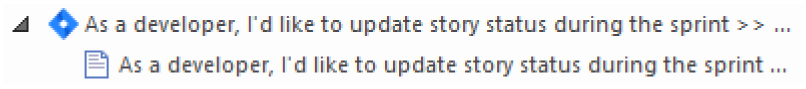
Link Multiple Items

Select multiple items to create multiple elements at the same time. All elements will be created with either the default type or «ExternalReference».

- To select a group of multiple items, select the first item, hold shift and select the last item.
- To select (or deselect) an individual item, hold Control and click the item.

Link an Existing Enterprise Architect Element

An existing element in Enterprise Architect can be linked to an existing item in the External Provider by dragging the local element from the Browser window and dropping it onto the external item. So in our example:


As a developer, I'd like to update story status during the sprint >> ...
As a developer, I'd like to update story status during the sprint ...

The Enterprise Architect element 'As a developer, I'd like to update story status during the sprint' was dragged from the Browser window and dropped onto the Jira item of the same name, creating a link between the two.

Create a Linked Item in the External Provider

Existing local elements in Enterprise Architect can be added to the External Provider in this way:

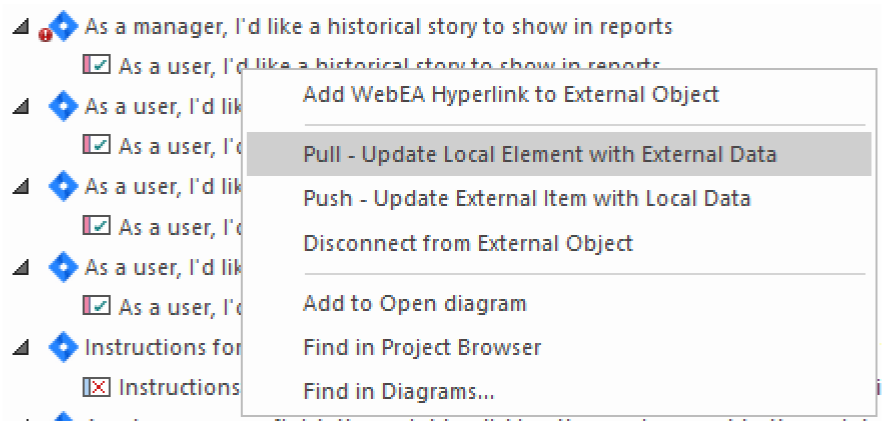
14. Select the local Enterprise Architect element (in the Browser window).
15. Browse the External Provider and select the required destination location.
16. In the 'External Data' menu, select 'Create Linked Item'.

Note:

- To create a new external item, the Integration Provider must be configured to allow this on the Pro Cloud Server (it is disallowed by default)
- Not all providers allow for creation of new items

Update Local Element ('Pull' changes)

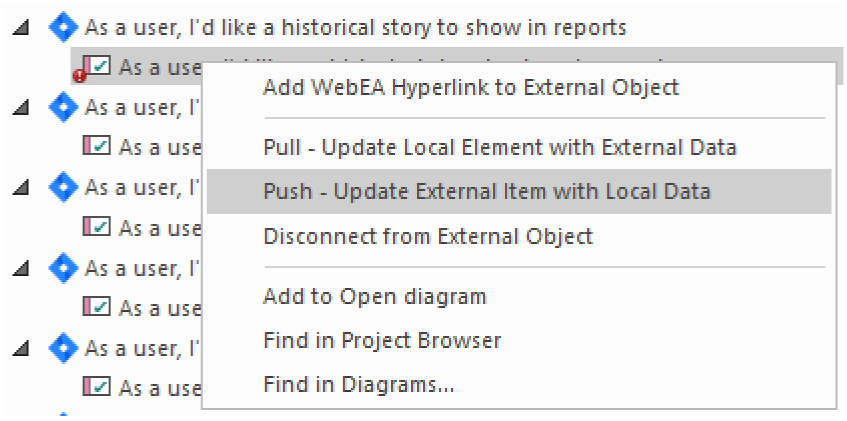
If either the local element or the external data item have been modified since the items were created or linked, an 'exclamation mark' indicator will display on top of the icon next to the name of the changed item. Right-click on the linked Enterprise Architect element and select the 'Pull - Update Local Element with External Data' option.


As a manager, I'd like a historical story to show in reports
As a user, I'd like a historical story to show in reports
Add WebEA Hyperlink to External Object
Pull - Update Local Element with External Data
Push - Update External Item with Local Data
Disconnect from External Object
Add to Open diagram
Find in Project Browser
Find in Diagrams...

Note: If the local element has been modified since the link was created then its changes will be overwritten with the external data.

Update External Item ('Push' changes)

If either the local element or the external data item have been modified since the items were created or linked, an 'exclamation mark' indicator will display on top of the icon next to the name of the changed item. Right-click on the Enterprise Architect element and select the 'Push - Update External Item with Local Data' option.


As a user, I'd like a historical story to show in reports
As a use
As a user, I'
Add WebEA Hyperlink to External Object
Pull - Update Local Element with External Data
Push - Update External Item with Local Data
Disconnect from External Object
Add to Open diagram
Find in Project Browser
Find in Diagrams...

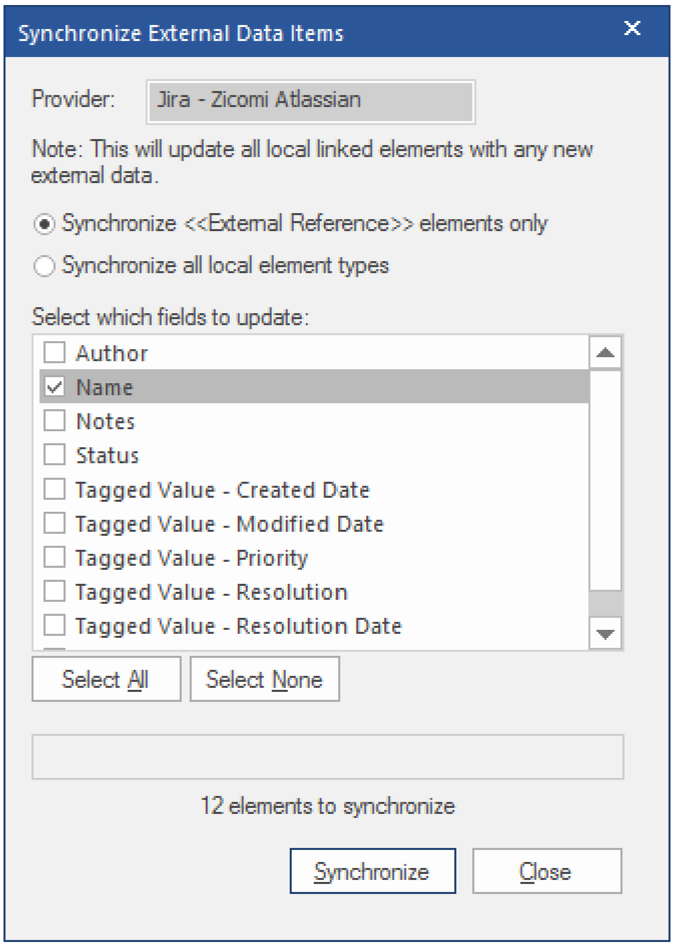
Note:

- If the external element has been modified since the link was created then its changes will be overwritten with the local data
- To update an external item the Integration Provider must be configured to allow this on the Pro Cloud Server (disallowed by default)

Synchronize All Local Elements

All local linked elements in Enterprise Architect can be quickly updated with new data from the linked items in the External Data Provider.

17. In the External Data window, display the data from the External Data Provider to be synchronized.
18. Click on the 'Hamburger' icon in the toolbar, and select the 'Synchronize Local Linked Elements' option.
19. On the 'Synchronize External Data Items' dialog, select the checkbox against each property field that should be updated with new data from the external item. Note that any local changes in the Enterprise Architect element in these fields will be overwritten with the external data.
20. The bottom of the window indicates how many local elements will be updated (note that it is unknown at this point if there have been any changes to the external item).
21. Click on the Synchronize button.


Synchronize External Data Items

Provider: Jira - Zicomi Atlassian

Note: This will update all local linked elements with any new external data.

● Synchronize <<External Reference>> elements only

○ Synchronize all local element types

Select which fields to update:

- ☐ Author
- ☑ Name
- ☐ Notes
- ☐ Status
- ☐ Tagged Value - Created Date
- ☐ Tagged Value - Modified Date
- ☐ Tagged Value - Priority
- ☐ Tagged Value - Resolution
- ☐ Tagged Value - Resolution Date

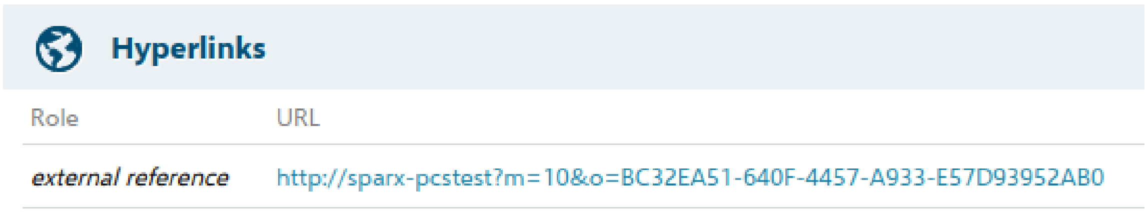
12 elements to synchronize

Item Hyperlinks to WebEA

The elements created from external items can be referenced from the external application via a hyperlink back to the element details hosted on WebEA. The benefits of using WebEA as the reference include:

- Easy access to the details via a web- browser
- Accesses the most up-to-date data for that element
- No need to install a local copy of Enterprise Architect

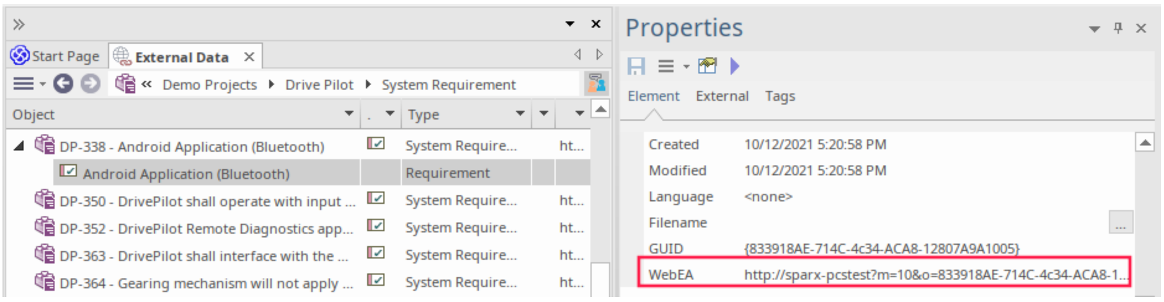
The hyperlink reference is set in the external item when a new element is created in Enterprise Architect. For example, here we have a hyperlink, inside an item in an external application, to a WebEA page for an element:



Hyperlinks

Role	URL
external reference	http://sparx-pcstest?m=10&o=BC32EA51-640F-4457-A933-E57D93952AB0

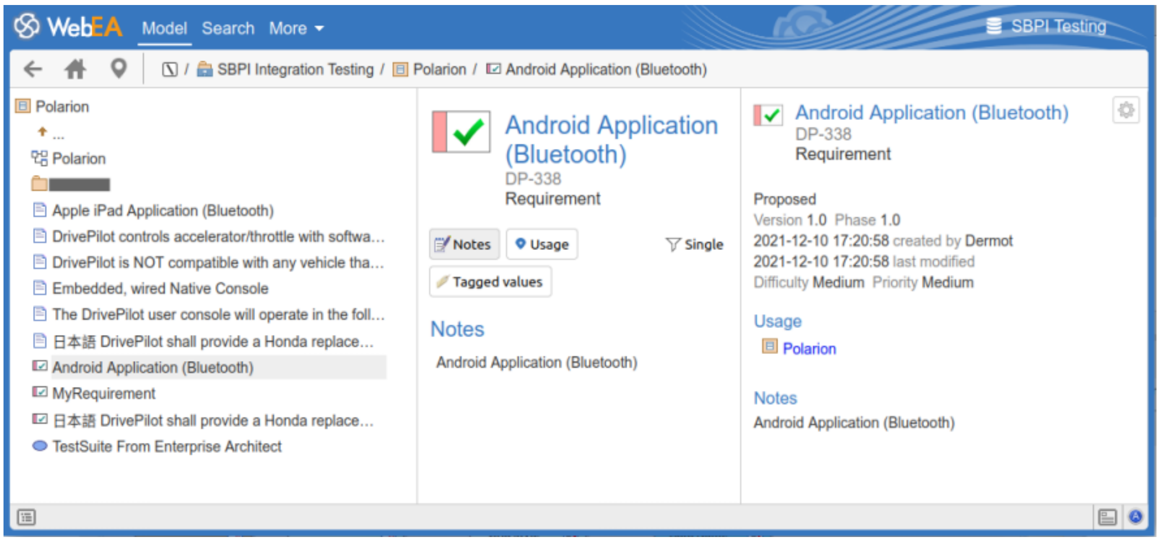
The hyperlink reference is system-generated and shown in the Properties window for that related element:



Object	.	Type	
DP-338 - Android Application (Bluetooth)		System Require...	ht...
Android Application (Bluetooth)		Requirement	
DP-350 - DrivePilot shall operate with input ...		System Require...	ht...
DP-352 - DrivePilot Remote Diagnostics app...		System Require...	ht...
DP-363 - DrivePilot shall interface with the ...		System Require...	ht...
DP-364 - Gearing mechanism will not apply ...		System Require...	ht...

Created	10/12/2021 5:20:58 PM
Modified	10/12/2021 5:20:58 PM
Language	<none>
Filename	
GUID	{833918AE-714C-4c34-ACA8-12807A9A1005}
WebEA	http://sparx-pcstest?m=10&o=833918AE-714C-4c34-ACA8-1...

Here is the WebEA page referenced from the external hyperlink:


WebEA
Model
Search
More
SBPI Testing
SBPI Integration Testing / Polarion / Android Application (Bluetooth)
Notes
Usage
Single
Tagged values

Polarion
...
Polarion
Apple iPad Application (Bluetooth)
DrivePilot controls accelerator/throttle with softwa...
DrivePilot is NOT compatible with any vehicle tha...
Embedded, wired Native Console
The DrivePilot user console will operate in the foll...
日本語 DrivePilot shall provide a Honda replace...
Android Application (Bluetooth)
MyRequirement
日本語 DrivePilot shall provide a Honda replace...
TestSuite From Enterprise Architect

Android Application (Bluetooth)
DP-338
Requirement

Notes
Android Application (Bluetooth)

Android Application (Bluetooth)
DP-338
Requirement

Proposed
Version 1.0 Phase 1.0
2021-12-10 17:20:58 created by Dermot
2021-12-10 17:20:58 last modified
Difficulty Medium Priority Medium

Usage
Polarion

Notes
Android Application (Bluetooth)

Supported

The Integrations that support hyperlink references to WebEA include:

- ALM
- Azure Dev Ops (TFS)
- Confluence (appended as 'info' field)

- Jira
- Jazz (including Doors NG)
- Polarion
- Wrike (as a comment on the item)

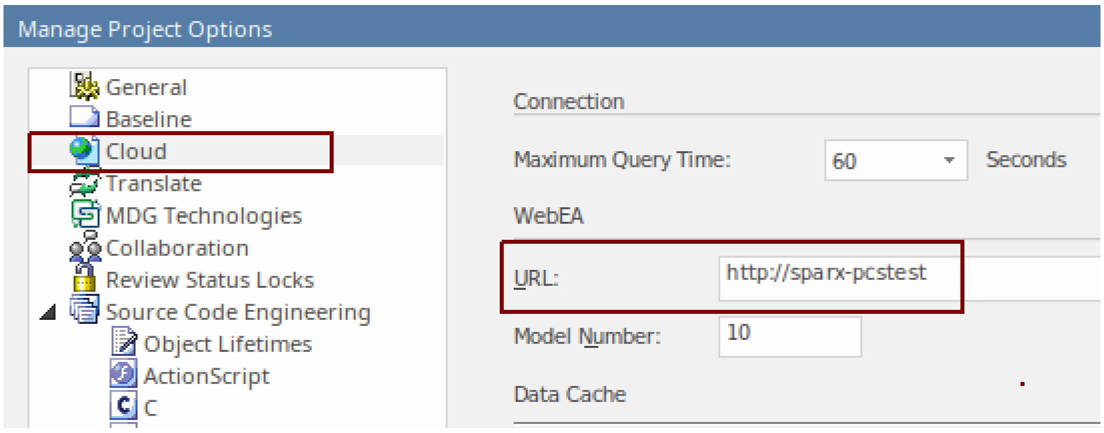
The external applications where this is not supported are:

- Dropbox
- Remote Enterprise Architect
- ServiceNow

Configuration

The configuration requires that WebEA is operating for that repository. To enable the external hyperlinks you must set the WebEA URL reference for that repository. The base reference is defined in:

- Settings > Model > Options > Cloud > URL



Maximum Query Time: 60 Seconds

URL: http://sparx-pcstest

Model Number: 10

For more details see the *Cloud Page* Help topic.

Writing a Custom Integration Plugin

We live in a highly connected world, and while there is a formidable list of out-of-the-box integrations provided with the Pro Cloud Server, custom (proprietary) integrations can be created with any product that has a standard web service interface. This facility will both open up the contents of the Enterprise Architect repository to an external tool and make the information in the external tool available within Enterprise Architect. For example, a Project Management tool might define work pages that would be useful to visualize in Enterprise Architect, or an automated testing tool could define test cases and test procedures that could be related to implementation and specification elements in Enterprise Architect. This will require some technical expertise to create an integration using one of a number of programming languages such as C++ or C# , but this only needs to be done once and can be used across any number of repositories.

The Pro Cloud Server and Enterprise Architect will do the heavy lifting, and there is no need for an administrator to change any of the base security settings for the Pro Cloud Server as the new integration will operate through the existing Ports and firewalls. There is also no need for the developer to write http listening code, allowing them to focus on determining and configuring the RESTful API calls to pass the external items' information into and out of the server.

When installing the Pro Cloud Server, enable the 'SBPI Examples' component to include the custom integration examples. When enabled, the default location for the example files is within the 'SBPI Examples\ExampleIntegrationPlugins' folder. For example:

C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI Examples\ExampleIntegrationPlugins

See the Pro Cloud Server Installation Help topic for more information.

Note, the 'SBPI Examples' installation option is not enabled by default. If you have already installed the Pro Cloud Server without the 'SBPI Examples', you can either perform a full re-install (enabling the 'SBPI Examples'), or use the installer's 'Change' option to add just the 'SBPI Examples' component.

To write your own Custom Integration Plug-in you can either start from scratch or make a copy of one of the examples and modify it. The Plug-ins can be written in either C++ or C#.

The examples are written using Visual Studio 2017 but this is not a pre-requisite.

The Custom Integration Plug-in must implement the interface defined in the ISBPIIntegrationPlugin, which is included in ISBPIIntegrationPlugin.h (for C++) or ISBPIIntegrationPlugin.cs (for C#).

The general flow of the program is:

- The user performs an action within Enterprise Architect that needs information from the Integration Plug-in
- The Plug-in will receive a call (or multiple calls) to the appropriate interface method
- The Plug-in parses the request and, if required, makes its own request to the actual data provider
- The Plug-in receives the result from the actual provider, and parses the data
- The Plug-in sends the response to Enterprise Architect via the provided callback functions; this can either be the actual data requested or an error value
- Enterprise Architect receives the callback data and displays it to the user

Class

Function/Class	Details
CheckVersion	(Not required in C#.) input: unsigned int version Returns true if your Plug-in supports the requested version. *Version 2 adds notifications when elements in Enterprise Architect are linked or unlinked to the external item, or when they are modified.* *Version 2 extends version 1, so returns true for versions up to and including the version supported.*

	e.g. return (version <= 2);

Create Plug-in	(Not required in C#.)
	The Plug-in must implement this export function:
	`extern "C" SBPI_API SBPI_INTEGRATION_PLUGIN CreatePlugin();`
	It must return a pointer to a class that implements the ISBPIIntegrationPlugin interface. The recommended implementation is:
	`SBPI_INTEGRATION_PLUGIN CreatePlugin()`
	`{`
	`    return new ExampleIntegrationPlugin;`
	`}`
	The newly created ISBPIIntegrationPlugin can be deleted when it receives the ISBPIIntegrationPlugin::Release method.

ISBPIIntegrationPlugin Interface	The dll Plug-in must implement all methods in the ISBPIIntegrationPlugin interface.

ISBPIIntegrationPlugin interface

The ISBPIIntegrationPlugin interface provides a range of methods.

ISBPIIntegrationPlugin Interface Methods

Most methods have a single string parameter (char* in C++, string in C#) that is encoded as JSON to provide a variety of parameters.

The ISBPIIntegrationPlugin interface provides these methods:

Method Name	Notes
CreateWebEAHyperlink	Adds a link to WebEA within the External Item.
GenericRequest	Reserved for future use.
GetAuthorisationDetails	Returns the Authorization methods that your provider requires. Supported types are: none, basic and OAuth.
GetDefaultFieldMapping	Returns a list of how the External Item fields and properties map to the Enterprise Architect element fields, properties and Tagged Values.
GetDefaultTypeMapping	Returns a list of how the External Item types map to the Enterprise Architect element types.
GetFields	Returns a list of known fields/properties in the External Provider.
GetIcon	Returns an icon for the External Provider.
GetItem	Returns the full properties for the selected item.
GetItemDiscussion	Populates the 'External' tab in the main Discussion window with comments and discussions on the selected external item.
GetItemList	On receipt of a request to populate the list of items in the External Data window for the selected menu level, this method fills in the items.
GetItemNotes	Populates the 'External' tab in the main Notes window with details of the selected external item.
GetItems	Returns the full properties for the selected items.
GetMenuList	On receipt of a request to populate the menu in the External Data window, this method fills in the menu items.
GetTypes	Returns a list of known types in the External Provider.
ItemLinked	(Requires Enterprise Architect Release 15.2 build 1559 and above.) Notification that an external item has been linked to an element in Enterprise Architect.

ItemUnlinked	(Requires Enterprise Architect Release 15.2 build 1559 and above.) Notification that an external item has been unlinked from an element in Enterprise Architect.
ItemUpdated	(Requires Enterprise Architect Release 15.2 build 1559 and above.) Notification that a linked element in Enterprise Architect has been updated or modified.
PostNewDiscussion	Adds a new comment or discussion to the External Item.
PostNewItem	Creates a new item in the External Provider.
PostOAuthCode	Makes an Access Token Request to exchange the code for an OAuth Access Token (using the OAuth token endpoint). Returns the accessToken and refreshToken with AddProperty.
PostUpdateItem	Updates the selected item in the External Provider.
PostUpdateItemNotes	Updates the notes of the selected item in the External Provider.
RefreshOAuthToken	Performs a refresh request against the OAuth refresh endpoint and returns the new accessToken and refreshToken.
Release	C++ only. This method is called by the controlling application when the interface Class is no longer required. The method deletes the Class created during the CreatePlugin() function.
SetAuthorisation	Receives authorization information that the user enters within Enterprise Architect.
SetCallbacks(const void ** callbackFunctions)	C++ only. Passes in an array of callback function pointers that are used by the Plug-in later on to pass data back to Enterprise Architect.
SetConfiguration	Receives the settings defined when the user sets up the Custom Integration Plug-in in Pro Cloud Server.

CreateWebEAHyperlink

Add a link to WebEA in the External Item.

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. Includes details of the WebEA link to be added.

Outputs via Callbacks

[Optional] LogMessage or SetError - to provide user feedback.

Details

This method receives a request to a WebEA link to the External Item.

Some providers allow for external links to be added to an item, while others might only allow it to be added as a comment or to the notes section.

The input parameters string contains information about the link to create, including:

- "itemID" - the ID of the item to be updated
- "webEALink" - the full URL of the WebEA link

Note that for the WebEA link to be valid, the 'Settings > Model > Options > Cloud' ribbon tab must have a valid WebEA base link specified.

GetAuthorisationDetails

Return the Authorization methods that your provider requires.

Supported types are:

- none
- basic
- OAuth

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. This is an empty string reserved for future use..

Outputs via Callbacks

- [Optional] AddProperty - to specify which authorisation methods are available and properties of these methods
- [Optional] LogMessage or SetError - to provide user feedback.

Details

For no authorisation, do nothing in this method.

Enterprise Architect only supports the OAuth 2.0 Authorization Code Grant type.

OAuth Authorisation will set Enterprise Architect to prompt to open a browser page for the user to log in to the OAuth provider (the 'authorization endpoint URI').

The OAuth provider will send back a 'code' to Enterprise Architect. Enterprise Architect will then call PostOAuthCode with the new code.

Note: If OAuth is specified then it will take priority and not allow basic authorisation.

OAuth requires the authorizationEndpointURIand redirectURI of the OAuth provider.

For Enterprise Architect, the redirectURI must be "http://localhost:8888/oauth/callback".

This usually needs to be added to your OAuth provider as an allowed redirect URI.

Basic Authorisation will set Enterprise Architect to prompt for a usename and password which will be passed back with each subsequent request.

Example Implementation

Basic Authorisation:

AddProperty(index, "basic", "true");

OAuth:

```
AddProperty(index, "OAuthConfiguration", "true");
AddProperty(index, "authorizationEndpointURI", "https://example.com/oauth/authorize");
AddProperty(index, "redirectURI", "http://localhost:8888/oauth/callback");
```

GetDefaultFieldMapping

Return a list of how the External Item fields and properties map to the Enterprise Architect element fields, properties and Tagged Values.

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. Reserved for future use.

Outputs via Callbacks

AddProperty - 2-3 values per field to map. Use a unique index value for each mapping:

- AddProperty(index, "externalField", "name");
- AddProperty(index, "internalField", "name");
- [Optional] AddProperty(index, "taggedValue", "My External Name");

[Optional] LogMessage or SetError - to provide user feedback.

Details

The values returned by this method are used to populate the Field Mapping in External Data Configuration.

The method defines the default values for how an External Item's fields/properties map to an Enterprise Architect element fields/properties/Tagged Values when linking an External Item.

The 'externalField' value should match the field ID as specified in GetFields, as well as the field names returned by, for example, GetItems.

The 'internalField' value should match an Enterprise Architect field name.

The options for internalField name are:

- 'Name'
- 'Alias'
- 'Author'
- 'Notes'
- 'Tagged Value' - This is a special case and requires AddProperty(index, "taggedValue", "tagged value name") to specify the name of the Tagged Value to use
- 'Status'
- 'Version'
- 'Phase'
- 'Keywords'
- 'Complexity'

- 'Scope'
- 'Multiplicity'
- 'Classifier'
- 'Abstract'
- 'Is Leaf'
- 'Language'
- 'Filename'
- 'Is Root'
- 'Is Specification'
- 'Persistence

Example:

- AddProperty(index, "externalField", "modifiedDate");
- AddProperty(index, "internalField", "Tagged Value");
- AddProperty(index, "taggedValue", "Modified Date");

GetDefaultTypeMapping

Return a default list of how the External Item types map to the Enterprise Architect element types.

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. Reserved for future use.

Outputs via Callbacks

AddProperty - 2-4 values per type to map. Use a unique index value for each mapping:

- AddProperty(index, "externalType", "Requirement");
- AddProperty(index, "internalType", "Requirement");
- [Optional] AddProperty(index, "stereotype", "document");
- [Optional] AddProperty(index, "toolbox", "Extended::Requirements");

[Optional] LogMessage or SetError - to provide user feedback.

Details

The values returned by this method are used to populate the Type Mapping in the External Data Configuration. It defines the default values for how an External Item's type maps to an Enterprise Architect element type when linking an item. This mapping is configurable by an end user in Enterprise Architect by editing the Type Mapping in the External Data Configuration.

The 'externalType' value should match the user-friendly Type name as specified in GetTypes, as well as the Type returned by, for example, GetItems.

The 'internalType' value should match an Enterprise Architect Type name.

Optionally, specify which Toolbox page this type came from. This will show when the user manually edits the mapping, allowing them to select another type from the same Toolbox page easily.

Always specify a default mapping type. This will be used if no matching mapping is found.

- AddProperty(index, "externalType", "Default");
- AddProperty(index, "internalType", "Requirement");

Optional stereotype:

- AddProperty(index, "stereotype", "document");

Specify the Toolbox page to allow users to change the mapping configuration to a type from the same Toolbox page.

- AddProperty(index, "toolbox", "Extended::Requirements");

Other mappings can be specified if required. For example:

- AddProperty(index, "externalType", "Feature");

- AddProperty(index, "internalType", "Feature");
- AddProperty(index, "toolbox", "Extended::Requirements");

GetFields

Return a list of known fields and properties in the External Provider.

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. Reserved for future use.

Outputs via Callbacks

AddProperty - 2 values per type to map. Use a unique index value for each mapping:

- AddProperty(index, "id", "type");
- AddProperty(index, "name", "Type");

[Optional] LogMessage or SetError - to provide user feedback.

Details

Most systems have an ID or non user-friendly name, as well as a user-friendly display name. Only Fields that are returned here will be recognized by Enterprise Architect.

The values returned here will be matched against the Type returned in DefaultFieldMapping and GetItem.

If possible, this list of fields should be dynamically created by querying the External Provider for a list of its known fields. The list of fields can be hard-coded for providers that do not provide a list of fields.

GetIcon

Return an icon for the External Provider.

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. Reserved for future use.

Outputs via Callbacks

- AddBinaryProperty - the binary data of the icon.
- [Optional] LogMessage or SetError - to provide user feedback.

Details

Specify an icon to be shown in Enterprise Architect External Data. The returned icon should be in png format, 20x16 pixels, with the left 4 columns of pixels blank and transparent.

Bit depth of 32 to include an alpha channel.

- AddBinaryProperty(index, <pointer to binary data array>, iconBytesSize);

GetItem

Return the full properties for the selected item.

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. Includes: • itemID - the unique id of the item (as passed back in GetItemList) • level - the level hierarchy of the requested menu (starting with 1 for first level) • currentID, currentName - the id and name of the current menu level • levelXID - the ID of the menu level X for all previous levels in the hierarchy; that is, level1ID, level2ID • levelXName - the name of the menu level X for all previous levels in the hierarchy; that is, level1Name, level2Name

Outputs via Callbacks

AddProperty - multiple calls per menu item:

- AddProperty(index, "id", "item1");
- AddProperty(index, "name", "Item 1");

Optional extra properties. These can be custom propertyID:propertyValue pairs, eg:

- AddProperty(index, "notes", "Example notes for item 1");
- AddProperty(index, "modifiedDate", "2022-04-06T11:33:44");
- AddProperty(index, "type", "Requirement");
- AddProperty(index, "url", "http://example.com/item/1");
- AddProperty(index, "propertyX", "Item 1 Property X");
- AddProperty(index, "propertyY", "Item 1 Property Y");
- AddProperty(index, "propertyZ", "Item 1 Property Z");

[Optional] LogMessage or SetError - to provide user feedback.

Details

This method receives a request to populate the 'External' tab in the main properties list with details about this external item.

This method should fill in the item's properties by calling the AddProperty callback method.

Each item requires two calls to AddProperty, and can accept extra optional calls to specify custom properties. The 'index' value is not required for this call. Set it to 0.

Mandatory properties:

- id - the id passed in specifies a unique id representing this menu item; the Plug-in can receive this id back in subsequent calls (such as when requesting the next sub-menu level)
- name - the user-displayable name of the menu item

Optional Properties:

The optional properties can be any propertyID:propertyValue pair. For the property to be listed in the Properties window, the property name must match a field value as returned by the GetFields method.

GetItemDiscussion

Return the discussions and comments for the selected item.

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. Includes: • itemID - the unique id of the item (as passed back in GetItemList)

Outputs via Callbacks

AddProperty - 3 calls are required per comment. Use a unique index value per comment:

- AddProperty(index, "authorName", "Example Person 1");
- AddProperty(index, "createdDate", "2022-04-06T12:34:56");
- AddProperty(index, "comment", "Example comment on Item 1 by Person 1");
- [Optional] AddProperty(index, "id", "0002"); // See note below
- [Optional] AddProperty(index, "parentID", "0001"); // See note below

[Optional] LogMessage or SetError - to provide user feedback.

Details

This method receives a request to populate the 'External' tab in the main Discussion Window with comments/discussion about this external item.

If an "id" property is added, this will be passed back if a user replies to a specific comment, allowing threaded discussions.

"parentID" will link this comment as a reply to the parent comment with the parentID

GetItemList

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. Includes: • level - the level hierarchy of the requested menu (starting with 1 for first level) • currentID, currentName - the id and name of the current menu level • levelXID - the ID of the menu level X for all previous levels in the hierarchy; that is, level1ID, level2ID • levelXName - the name of the menu level X for all previous levels in the hierarchy; that is, level1Name, level2Name

Outputs via Callbacks

[Optional] AddProperty - multiple calls per item:

- AddProperty(index, "id", "myitemid");
- AddProperty(index, "name", "Example Item 3");
- AddProperty(index, "modifiedDate", "2022-04-06T11:33:44");
- AddProperty(index, "type", "Requirement");
- AddProperty(index, "url", "http://example.com/item/3");

[Optional] LogMessage or SetError - to provide user feedback.

Details

This method receives a request to populate the list of items in the External Data window for the selected menu level. This method should fill in the items by calling the AddProperty callback method.

Each item requires five calls to AddProperty to fill in the details of the item in the list. And each item must use a unique 'index' value as the first parameter.

- id - the id passed in specifies a unique id representing this item; the Plug-in can receive this id back in subsequent calls (such as when requesting item details or notes)
- name - the user-displayable name of the item
- modifiedDate - the date/time that the item was last modified, in the format: YYYY-MM-DDTHH:MM:SS
- type - the type of the item as understood by the external provider; for example Defect, Task, Requirement, Document
- url - a url that will take a user directly to this item; this is used by Enterprise architect when selecting 'Open External Item in Browser Window'

Don't return an error for a blank response. Simply do nothing.

GetItemNotes

Return the notes or descriptions for the selected item.

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. Includes: • itemID - the unique id of the item (as passed back in GetItemList)

Outputs via Callbacks

- AddProperty - notes - the text value of the notes for the item; some basic HTML markup is allowed
- [Optional] LogMessage or SetError - to provide user feedback.

Details

This method receives a request to populate the 'External' tab in the main Notes Window with details about this external item.

GetItems

Return the full properties for the requested items.

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. Includes: • itemIDs - a comma separated list of item IDs • level - the level hierarchy of the requested menu (starting with 1 for first level) • currentID, currentName - the ID and name of the current menu level • levelXID - the ID of the menu level X for all previous levels in the hierarchy; that is, level1ID, level2ID • levelXName - the name of the menu level X for all previous levels in the hierarchy; that is level1Name, level2Name

Outputs via Callbacks

AddProperty - multiple calls per menu item:

- AddProperty(index, "id", "item1");
- AddProperty(index, "name", "Item 1");

Optional extra properties. These can be custom propertyID:propertyValue pairs, eg:

- AddProperty(index, "notes", "Example notes for item 1");
- AddProperty(index, "modifiedDate", "2022-04-06T11:33:44");
- AddProperty(index, "type", "Requirement");
- AddProperty(index, "url", "http://example.com/item/1");
- AddProperty(index, "propertyX", "Item 1 Property X");
- AddProperty(index, "propertyY", "Item 1 Property Y");
- AddProperty(index, "propertyZ", "Item 1 Property Z");

[Optional] LogMessage or SetError - to provide user feedback.

Details

Return the properties for a list of items. Recommended: If the provider has a way of querying a list of items, then it is recommended to run a single query and return the results. If the provider can't query multiple items, then run multiple individual 'GetItem' calls and concatenate the results.

The returned values should be the same as for GetItem, but specify a unique 'index' value for each different item.

GetMenuList

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. Includes: • level - the level hierarchy of the requested menu (starting with 1 for first level) • currentID, currentName - the ID and name of the current menu level • levelXID - the id of the menu level X for all previous levels in the hierarchy; that is, level1ID, level2ID • levelXName - the name of the menu level X for all previous levels in the hierarchy; that is level1Name, level2Name

Outputs via Callbacks

- AddProperty - 2 calls per menu item: "id", "name".
- [Optional] LogMessage or SetError - to provide user feedback.

Details

This method receives a request to populate the menu in the External Data window. Each level in the menu hierarchy will trigger a new request, and the parameters will include the level being requested.

This method should fill in the menu items by calling the AddProperty callback method.

Each item in the menu requires two calls to AddProperty. Each item must use a unique 'index' value as the first parameter.

- id - the id passed in specifies a unique id representing this menu item; the Plug-in could receive this id back in subsequent calls (such as when requesting the next sub-menu level)
- name - the user-displayable name of the menu item

Don't return an error for a blank response. Simply do nothing.

GetTypes

Return a list of known types in the External Provider.

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. Reserved for future use.

Outputs via Callbacks

AddProperty - 2 values per type to map. Use a unique index value for each mapping:

- AddProperty(index, "id", "artifact");
- AddProperty(index, "name", "Artifact");

[Optional] LogMessage or SetError - to provide user feedback.

Details

Most systems have an ID or non user-friendly name, as well as a user-friendly display name. Only Types that are returned here will be recognized by Enterprise Architect.

The values returned here will be matched against the Type returned in DefaultTypeMapping and GetItem.

If possible, this list of types should be dynamically created by querying the External Provider for a list of its known types. The list of types can be hard-coded for providers that do not provide a list of types.

ItemLinked

(Requires Enterprise Architect Release 15.2 build 1559 and above.)

Notification that an external item has been linked to an element in Enterprise Architect.

Inputs

parameters	• C++: const char* • C#: string A JSON string of parameters. • itemID - the unique ID of the item (as passed back in GetItemList)

Outputs via Callbacks

[Optional] LogMessage or SetError - to provide user feedback.

Details

This method is called whenever an external item is linked to an element in Enterprise Architect - either creating a new element or linking to an existing element.

This method is part of version 2 of the ISBPIIntegrationPlugin interface. For C++ Plug-ins ensure that CheckVersion handles version 2 correctly.

ItemUnlinked

(Requires Enterprise Architect Release 15.2 build 1559 and above.)

Notification that an external item has been unlinked from an element in Enterprise Architect.

Inputs

parameters	• C++: const char* • C#: string A JSON string of parameters. • itemID - the unique ID of the item (as passed back in GetItemList) • eaElementGUID - the GUID of the Enterprise Architect element

Outputs via Callbacks

[Optional] LogMessage or SetError - to provide user feedback.

Details

This method is called whenever an element in Enterprise Architect is unlinked from the external item. This can be due to the element being deleted, or through the menu option 'Disconnect from External Object'.

This method is part of version 2 of the ISBPIIntegrationPlugin interface. For C++ Plug-ins ensure that CheckVersion handles version 2 correctly.

ItemUpdated

(Requires Enterprise Architect Release 15.2 build 1559 and above.)

Notification that a linked element in Enterprise Architect has been updated or modified.

Inputs

parameters	• C++: const char* • C#: string A JSON string of parameters. • itemID - the unique ID of the item (as passed back in GetItemList)

Outputs via Callbacks

[Optional] LogMessage or SetError - to provide user feedback.

Details

This method is called whenever a linked element in Enterprise Architect is modified; for example, name change, property modified, notes updated.

This method is part of version 2 of the ISBPIIntegrationPlugin interface. For C++ Plug-ins ensure that CheckVersion handles version 2 correctly.

PostNewDiscussion

Add a new comment/discussion to the External Item

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters, including details of the new comment to be added.

Outputs via Callbacks

[Optional] LogMessage or SetError - to provide user feedback.

Details

This method receives a request to create a new discussion/comment in the External Item. The input parameters string contains information about the comment to create, including:

- "itemID" - the ID of the item to be updated
- "author"
- "comment"
- [optional] parentID - the ID of the parent comment when using threaded comments; this is the ID that was passed back in GetItemDiscussion.

PostNewItem

Creates a new item in the External Provider.

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. Includes details of the new item to be created.

Outputs via Callbacks

AddProperty - returns the ID of the newly created item. Enterprise Architect will not consider the creation successful unless a valid ID is returned.

- AddProperty(0, "id", "itemX");

[Optional] LogMessage or SetError - to provide user feedback.

Details

This method receives a request to create a new item in the External Provider. The input parameters string contains information about the item to create, including:

- "title"
- "type"
- "stereotype"
- "description"

PostOAuthCode

Inputs

Parameter	Details
code	• C++: const char* • C#: string The OAuth code value returned to the user after successful authorization.

Outputs via Callbacks

- AddProperty - "accessToken" to be used for OAuth
- [Optional] AddProperty - "refreshToken" for OAuth
- [Optional] LogMessage or SetError - to provide user feedback.

Details

After a successful login to the OAuth provider by the user, Enterprise Architect will call this method with the OAuth 'code'. Make an Access Token Request to exchange the code for an OAuth Access Token (using the OAuth token endpoint).

Return the accessToken and refreshToken with AddProperty.

PostUpdateItem

Updates the selected item in the External Provider. The Notes of the selected item can be updated by the PostUpdateItemNotes method.

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. Includes details of the item to be updated.

Outputs via Callbacks

[Optional] LogMessage or SetError - to provide user feedback.

Details

This method receives a request to update an item in the External Provider. The input parameters string contains information about the item to create, including:

- "itemID" - the ID of the item to be updated
- "title"
- "type"
- "stereotype"
- "description"

Note: the new values might not actually be different to the current values.

PostUpdateItemNotes

Updates the notes of the selected item in the External Provider.

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters that include details of the item to be updated.

Outputs via Callbacks

[Optional] LogMessage or SetError - to provide user feedback.

Details

This method receives a request to update the notes of an item in the External Provider. The input parameters string contains information about the item to create, including:

- "itemID" - the ID of the item to be updated
- "notes"

RefreshOAuthToken

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string The OAuth refresh token to be used to get a fresh access token.

Outputs via Callbacks

- AddProperty - "accessToken" to be used for OAuth
- [Optional] AddProperty - "refreshToken" for OAuth
- [Optional] LogMessage or SetError - to provide user feedback.

Details

If any method here returns a 401 (Unauthorized) via SetErrorCode then Enterprise Architect will attempt to use the refresh token (if it was supplied) and call this method.

Perform a refresh request against the OAuth refresh endpoint and return the new accessToken and refreshToken.

SetAuthorisation

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. Available fields are: • username • password • accessToken - OAuth access token

Outputs via Callbacks

[Optional] LogMessage or SetError - to provide user feedback.

Details

This method receives authorisation information that the user enters within Enterprise Architect.

It consists of either username:password credentials or an OAuth access token.

Enterprise Architect will call SetAuthorisation at the start of each session (e.g. open System Integration, open a model, or change user). Use the values provided for any calls to the external provider.

Do not store these as 'static' in any way. Storing as simple members is OK as a new instance of this Class will be created for each new session.

Example Implementation

```
void ExampleIntegrationPlugin::SetAuthorisation(const char* parameters)
{
	LogMessage(LOG_TRACE, __FUNCTION__);

	Json::Value jsonParameters;
	if (strlen(parameters))
	{
		std::stringstream(parameters) >> jsonParameters;
	}

	m_username = jsonParameters["username"].asString();			// If basic authorisation is used.
	m_password = jsonParameters["password"].asString();
```

```
        m_accessToken = jsonParameters["accessToken"].asString();  // If OAuth is used.
}
```

SetCallbacks

C++ only. This only needs to be implemented in C++. The code in the example is sufficient and doesn't need to be modified. Extra error or bounds checking can be added.

Inputs

Parameter	Details
const void ** callbackFunctions	An array of callback function pointers to be used to pass data back to Enterprise Architect.

Outputs via Callbacks

None

Details

C++ Plug-ins will receive this method soon after creation. It passes in an array of callback function pointers which are used by the Plug-in later on to pass data back to Enterprise Architect

Example Implementation

```
void ExampleIntegrationPlugin::SetCallbacks(const void ** callbackFunctions)
{
        if (callbackFunctions)
        {
                AddProperty = (AddPropertyPtr)callbackFunctions[0];
                AddBinaryProperty = (AddBinaryPropertyPtr)callbackFunctions[1];
                SetErrorCode = (SetErrorCodePtr)callbackFunctions[2];
                SetError = (SetErrorPtr)callbackFunctions[3];
                LogMessage = (LogMessagePtr)callbackFunctions[4];
        }
}
```

SetConfiguration

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. See Details for more information about available parameters.

Outputs via Callbacks

[Optional] LogMessage - set log messages about the configuration settings received. Be careful not to log sensitive information.

Note: SetError callbacks will be ignored for this method.

Details

This method receives the details that the user inputs into the Pro Cloud Server configuration when enabling this Custom Integration Plug-in.

It includes these details:

External Server to connect to:

- serverName
- serverPort
- serverProtocol
- baseURL - the url folder to be appended to the url

The server settings combine to form a URL as such: <protocol>://<serverName>:<serverPort>/<baseURL>.

Hardcoded Credentials - These are optional and can be used to connect to a provider with a generic account:

- username
- password

Permissions - sets whether users can perform the specified actions on the external provider:

- allowCreateItems
- allowModifyItems
- allowPostDiscussions

Proxy settings:

- proxyServer
- proxyBypass
- proxyUsername
- proxyPassword

Example Implementation

```
void ExampleIntegrationPlugin::SetConfiguration(const char* parameters)
{
	LogMessage(LOG_TRACE, __FUNCTION__);

	Json::Value jsonParameters;
	if (strlen(parameters))
	{
		std::stringstream(parameters) >> jsonParameters;
	}

	// Store the settings as member variables for later use.
	m_serverName = jsonParameters["serverName"].asString();
	m_serverPort = jsonParameters["serverPort"].asString();
	m_serverProtocol = jsonParameters["serverProtocol"].asString();
	m_baseURL = jsonParameters["baseURL"].asString();

	m_settingsUsername = jsonParameters["username"].asString();
	m_settingsPassword = jsonParameters["password"].asString();

	m_allowCreateItems = jsonParameters["allowCreateItems"].asString();
	m_allowModifyItems = jsonParameters["allowModifyItems"].asString();
	m_allowPostDiscussions = jsonParameters["allowPostDiscussions"].asString();

	m_proxyServer = jsonParameters["proxyServer"].asString();
	m_proxyBypass = jsonParameters["proxyBypass"].asString();
	m_proxyUsername = jsonParameters["proxyUsername"].asString();
	m_proxyPassword = jsonParameters["proxyPassword"].asString();
}
```

Custom SBPI Services

Custom SBPI Services are user-defined integration Plug-ins that can be invoked by Enterprise Architect's scripting or Add-ins, which can then use the responses to manipulate the repository data. The Custom Plug-In can be called from Enterprise Architect using the Repository**.**CallSBPI automation interface method.

The scope of what can be requested of the service and what it can return to Enterprise Architect is not limited. For example, the Custom Service could respond to a request to gather information from a third-party service, and return it to the Enterprise Architect script. Other possible uses include performing actions on the model data itself via OSLC calls, or running custom-built processes on model data.

Benefits

- Allow arbitrary requests and responses
- Once configured for a model, can be called from scripting or Add-Ins
- Lifetime and request forwarding automatically handled by Pro Cloud Server
- Can be written in multiple programming languages, including C++ or C#

Overview

To write your own Custom Service Plug-in you can either start from scratch or make a copy of one of the examples and modify it.

When installing the Pro Cloud Server, enable the 'SBPI Examples' component to include the custom service examples. When enabled, the default location for the example files is within the 'SBPI Examples\ExampleServicePlugins' folder. For example:

C:\Program Files (x86)\Sparx Systems\Pro Cloud Server\SBPI Examples\ExampleServicePlugins

See the Pro Cloud Server Installation Help topic for more information.

Note, the 'SBPI Examples' installation option is not enabled by default. If you have already installed the Pro Cloud Server without the 'SBPI Examples', you can either perform a full re-install (enabling the 'SBPI Examples'), or use the installer's 'Change' option to add just the 'SBPI Examples' component.

The Plug-ins can be written in either C++ or C#.

The examples are written using Visual Studio 2017, but this is not a pre-requisite.

The Custom Service Plug-in must implement the interface defined in ISBPIServicePlugin, which is included in ISBPIServicePlugin.h (for C++) or ISBPIServicePlugin.cs (for C#).

The general flow of the program is:

- The user performs an action within Enterprise Architect that needs information from the Service Plug-in via the CallSBPI automation interface
- The Plug-in receives the request
- The Plug-in parses the request and performs any action required (for example, call third-party service, run program, make OSLC calls to Pro Cloud Server)
- The Plug-in sends a response to Enterprise Architect via the provided callback functions; this can either be the actual data requested or an error value
- Enterprise Architect receives the callback data and uses it in the script or Add-in

Interface

Function/Class	Details

Create Plug-in (not required in C#)	The Plug-in must implement this export function:
	`extern "C" SBPI_SERVICE_API SBPI_SERVICE_PLUGIN CreatePlugin();`
	It must return a pointer to a Class that implements the ISBPIServicePlugin interface. The recommended implementation is:
	`SBPI_SERVICE_PLUGIN CreatePlugin()`
	`{`
	`    return new ExampleServicePlugin;`
	`}`
	The newly created ISBPIServicePlugin can be deleted when it receives the ISBPIServicePlugin::Release method.

ISBPIServicePlugin interface	The dll Plug-in must implement all methods in the ISBPIServicePlugin interface.

ISBPIServicePlugin interface

ISBPIServicePlugin interface methods

Method Name	Notes
Release	C++ only. This is called by the controlling application when the interface class is no longer required. The method should delete the class created during the CreatePlugin() function.
SetCallbacks	C++ only. Passes in an array of callback function pointers that are used by the Plug-in later on to pass data back to Enterprise Architect.
SetConfiguration	Receives the settings defined when the user sets up the Custom Service Plugin in Pro Cloud Server.
HandleRequest	Generic request from Enterprise Architect. The plug-in can perform any action required and return data or error-codes back to Enterprise Architect via the callback methods.

HandleRequest

Inputs

method	• C++: const char* • C#: string The name of the method to perform. This allows the plugin to make a choice on what action to perform without having to parse the parameters string fully first.
parameters	• C++: const char* • C#: string A JSON string of parameters. See Details for more information about available parameters.

Outputs via Callbacks

[Optional] Result, LogMessage or SetError - to provide user feedback.

Details

This is the main function to the plug-in. It should handle all incoming 'method' requests and perform the action required, and return any data via the 'Result' callback.

Example Implementation

```
void ExampleServicePlugin::HandleRequest(const char* method, const char* parameters)
{
   LogMessage(LOG_TRACE, std::string(__FUNCTION__ + " - Method = "s + method).c_str());

   // This example method demonstrates how to extract various parameter types from parameters.
   // This is done here with jsoncpp library but can be done with any compliant JSON library.
   Json::Value jsonParameters;
   if (strlen(parameters))
   {
      std::stringstream(parameters) >> jsonParameters;
   }

   if (std::string(method) == "DoSomething")
   {
      int myNumber = jsonParameters["myNumber"].asInt();
```

```
        double myFloat = jsonParameters["myFloat"].asDouble();
        std::string myString = jsonParameters["myString"].asString();
        std::list<int> myArrayOfNumbers;
        for (auto& myValue : jsonParameters["myArrayOfNumbers"])
        {
           myArrayOfNumbers.push_back(myValue.asInt());
        }
        std::list<std::string> myArrayOfStrings;
        for (auto& myValue : jsonParameters["myArrayOfStrings"])
        {
           myArrayOfStrings.push_back(myValue.asString());
        }

        std::string result = "Example User SBPI Service Plugin in C++. DoSomething received parameters: myNumber = "
+ std::to_string(myNumber)
           + ", myfloat = " + std::to_string(myFloat)
           + ", myString = " + myString;

        // Set the result string.
        Result(result.c_str());
     }
     else if (std::string(method) == "DoSomethingToElement")
     {
        // This example method demonstrates how to respond to a user performing a task on a single element.
        std::string elementID = jsonParameters["elementGUID"].asString();

        Result(std::string("Example User SBPI Plugin in C++. DoSomething to element with GUID: " +
elementID).c_str());
     }
     else
     {
        // Set an error string
        SetError(std::string("Unknown method: " + std::string(method)).c_str());
     }
}
```

SetCallbacks

C++ only. This only needs to be implemented in C++. The code in the example is sufficient and doesn't need to be modified. Extra error or bounds checking can be added.

Inputs

Parameter	Details
const void ** callbackFunctions	An array of callback function pointers to be used to pass data back to Enterprise Architect.

Outputs via Callbacks

None

Details

C++ Plug-ins will receive this method soon after creation. It passes in an array of callback function pointers which are used by the Plug-in later on to pass data back to Enterprise Architect

Example Implementation

```
void ExampleServicePlugin::SetCallbacks(const void ** callbackFunctions)
{
    if (callbackFunctions)
    {
        Result = (ResultPtr)callbackFunctions[0];
        SetErrorCode = (SetErrorCodePtr)callbackFunctions[1];
        SetError = (SetErrorPtr)callbackFunctions[2];
        LogMessage = (LogMessagePtr)callbackFunctions[3];
    }
}
```

SetConfiguration

Inputs

Parameter	Details
parameters	• C++: const char* • C#: string A JSON string of parameters. See Details for more information about available parameters.

Outputs via Callbacks

[Optional] LogMessage - set log messages about the configuration settings received. Be careful not to log sensitive information.

Note: SetError callbacks will be ignored for this method.

Details

This method receives the details that the user inputs into the Pro Cloud Server configuration when enabling this Custom Integration Plug-in.

It includes these details:

- Custom item/value pairs of settings entered in 'Custom Properties' section.

Proxy settings:

- proxyServer
- proxyBypass
- proxyUsername
- proxyPassword

Example Implementation

```
void ExampleServicePlugin::SetConfiguration(const char* parameters)
{
    LogMessage(LOG_TRACE, std::string(__FUNCTION__).c_str());

    Json::Value jsonParameters;
    if (strlen(parameters))
    {
        std::stringstream(parameters) >> jsonParameters;
    }
    for (auto& myProperty : jsonParameters.getMemberNames())
    {
        m_properties[myProperty] = jsonParameters[myProperty].asString();
    }
}
```

Example Script

This JavaScript script details how to send a simple request to a Custom Service plugin:

```
!INC Local Scripts.EAConstants-JavaScript

/*
* Script Name: Custom Service Example
* Author: Sparx Systems
* Purpose: Demonstrate the use of the SBPI automation interface for Custom Service plugins
* Date: 2022-02-28
*/

// Sends a simple request to the plugin with some parameters.
function SimpleRequest()
{
	// Show the script output window
	Repository.EnsureOutputVisible( "Script" );
	Session.Output("JavaScript Custom Plugin EXAMPLE");
	Session.Output("======================================");

	// Send data with the request by adding parameters using InsertSBPIParameter.
	var packedParameters = '';

	// Optional data to send with extra parameters
	packedParameters = Repository.InsertSBPIParameter(packedParameters, 'myNumber', 25);
	packedParameters = Repository.InsertSBPIParameter(packedParameters, 'myFloat', 123.456);
	packedParameters = Repository.InsertSBPIParameter(packedParameters, 'myString', 'Hello World');

	Session.Output("Sending simple request to plugin to 'DoSomething' method");
	var response = SBPIRequest('csvc', 'DoSomething', packedParameters);
}

// Helper function to send a request to the Custom plugin and check for errors.
function SBPIRequest(prefix, method, packedParameters)
{
	// Specify the prefix of the plugin. This is configured in the Pro Cloud Config Client.
	var response = Repository.CallSBPI(prefix, method, packedParameters);
	if (response == '')
	{
		Session.Output('Error from plugin: ' + Repository.GetLastError());
		}
		else
		{
			Session.Output('Success: ' + response);
		}

		return response;
}

function main()
{
		// Sends a simple request to the plugin with some parameters.
		SimpleRequest();
}

main();
```