



# User Guide

# ARCAD-API

Version 24.0

**Publication Date: January, 2024**

*Prepared by the ARCAD Documentation Team*



#### North America & LATAM

1 N. State St, 15th Floor  
Chicago, IL  
USA  
1-603-371-9074  
1-603-371-3256 (support calls only)  
sales-us@arcadsoftware.com

#### EMEA (HQ)

55 Rue Adrastée – Parc Altaïs  
74650 Chavanod/Annecy  
France  
+33 450 578 396  
sales-eu@arcadsoftware.com

#### Asia Pacific

5 Shenton Way #22-04  
UIC Building  
Singapore 068808  
sales-asia@arcadsoftware.com

**Copyright © 1992-2024 by ARCAD. All rights reserved.**

The following terms are names owned by International Business Machines Corporation in the United States, other countries, or both: AS/400®, ClearCase, ClearQuest®, DB2, DB2 Connect™, DB2 Universal Database™, ibm.com, IBM i, iSeries, System i, OS/400, Rational®, SP2, Service Pack, WebSphere. Java and all names based on Java are owned by Oracle Corp. in the United States, other countries, or both. Eclipse is a registered trademark of Eclipse Foundation, Inc. Other names of companies, products or services are the property of their respective owners.

# Contact ARCAD

Headquartered in France at the foot of the Alps, ARCAD offers global services and has offices and partners all over the world. ARCAD partners with leading-edge companies throughout the world to offer full services, close to home.

Visit our website to [Contact Us](#) and find out more about our company and partners, or to request a demo.

The [ARCAD Customer Portal](#) is intended for current and potential customers that have full or trial versions of ARCAD software. If you already use or are interested in using an ARCAD product, the portal lets you view all of your current licenses and generate your own temporary license keys for most ARCAD products. It grants you access to the ARCAD product knowledge base (new releases, release notes and current documentation).

Do you have a request for change or have you encountered a bug? Log into the [ARCAD Helpdesk](#) and create a ticket.

ARCAD guarantees consultant support 24 hours a day, 5 days a week (24/5) to registered members. Calls received are redirected, according to the hour, to put you in contact with a support team in or near your timezone.

Country	Address	Account Contact	Support Contact
France	ARCAD Software (HQ) 55 Rue Adrastée 74650 Chavanod		Worldwide 24/7: +1 603 371 3256  France only: +33 450 57 28 00  <a href="mailto:support@arcadsoftware.com">support@arcadsoftware.com</a>  <a href="#">ARCAD Helpdesk</a>
	ARCAD Software 17 chemin de la plaine 07200, Saint-Didier-sous-Aubenas		
Germany	ARCAD Software Deutschland GmbH c/o Pramex International GmbH Savignystr. 43, 60325 Frankfurt am Main		
China	ARCAD Software #2035, Yuehai Plaza, 180 Wanbo 2nd Road, Nancun, Panyu District, Canton		
India	ARCAD Software D-280/281/282, Vibhuti Khand Gomti Nagar, Lucknow	+86 (020)22324643 +86 (020)22324649 <a href="mailto:sales-asia@arcadsoftware.com">sales-asia@arcadsoftware.com</a>	
Singapore	ARCAD Software 5 Shenton Way #22-04 UIC Building Singapore 068808		
USA	ARCAD Software 1 N. State St, 15th Floor Chicago, IL	+1 (603) 371-9074 +1 (603)-371-3256 (support calls only) <a href="mailto:sales-us@arcadsoftware.com">sales-us@arcadsoftware.com</a>	

Table 1: Contact ARCAD

# Preface

---

## Document purpose

This document is intended to guide you through using ARCAD-API.

## Intended Audience

This document is intended for all ARCAD-API users.

## Related Documentation

ARCAD technical documentation can be accessed from the product's online help or by logging into the [Customer Portal](#) on our website.

Related Documentation
Release Notes
ARCAD Glossary

*Table 2: Related Documentation*

Unless stated otherwise, all content is valid for the most current version of ARCAD-API listed as well as every subsequent version.

Product Version	Document Version	Publication Date	Update Record
≥ 24.0	1.7	January, 2024	No functional changes.
23.0	1.6	January, 2023	No functional changes.

*Table 3: ARCAD-API User Guide Publication Record*

# Contents

---

<b>Contact ARCAD</b> .....	<b>3</b>
<b>Preface</b> .....	<b>4</b>
<b>Contents</b> .....	<b>5</b>
<b>Tables</b> .....	<b>10</b>
<b>Figures</b> .....	<b>11</b>

## ABOUT ARCAD-API

---

<b>1 About ARCAD-API</b> .....	<b>14</b>
<b>2 Main concepts</b> .....	<b>17</b>
<b>3 Overview</b> .....	<b>18</b>

## SYSTEM REQUIREMENTS

---

<b>4 System requirements for the ARCAD-API Studio</b> .....	<b>20</b>
4.1 Windows.....	20
4.1.1 Hardware requirements.....	20
4.1.2 Software requirements.....	20
4.2 Linux.....	20
4.2.1 Hardware requirements.....	21
4.2.2 Software requirements.....	21
<b>5 System requirements for the ARCAD-API Server</b> .....	<b>22</b>
5.1 Windows.....	22
5.1.1 Hardware requirements.....	22
5.1.2 Software requirements.....	22
5.2 Linux.....	23
5.2.1 Hardware requirements.....	23
5.2.2 Software requirements.....	23
5.3 IBM i.....	24
5.3.1 Hardware requirements.....	24
5.3.2 Software requirements.....	24
5.3.3 Host name resolution.....	24
5.3.4 Users for the ARCAD-API Server.....	25

## INSTALLATION

---

<b>6 Installing the ARCAD-API Studio</b> .....	<b>27</b>
6.1 Prerequisites.....	27
6.2 Windows.....	27
6.2.1 Install.....	27
6.2.2 Update.....	28
6.2.3 Uninstall.....	29
6.3 Linux.....	29

6.3.1 Install.....	29
6.3.2 Update.....	30
6.3.3 Uninstall.....	30
<b>7 Installing the ARCAD-API Server.....</b>	<b>31</b>
7.1 Prerequisites.....	31
7.2 Windows.....	31
7.2.1 Install.....	31
7.2.2 Update.....	33
7.2.3 Uninstall.....	34
7.3 Linux.....	34
7.3.1 Install.....	34
7.3.2 Update.....	35
7.3.3 Uninstall.....	36
7.4 IBM i.....	37
7.4.1 Install.....	37
7.4.2 Update.....	38
7.4.3 Uninstall.....	39

## CONFIGURING ARCAD-API

---

<b>Introduction to ARCAD-API configuration.....</b>	<b>42</b>
<b>8 Managing the ARCAD-API Server.....</b>	<b>43</b>
8.1 Creating an ARCAD-API Server connection.....	43
8.2 Connecting to the ARCAD-API Server.....	44
8.3 Configuring the TLS settings.....	45
8.4 Configuring the ARCAD-API Server.....	46
8.5 Managing the ARCAD-API Server preferences.....	46
8.6 Viewing ARCAD-API Server logs.....	46
8.7 Verifying ARCAD-API Server versions.....	47
<b>9 Configuring the ARCAD-API Server.....</b>	<b>48</b>
9.1 Defining the APIGEE Edge Settings.....	48
9.2 Defining the API Gateway General Settings.....	49
9.3 Defining the Web Service Descriptors Storage Area.....	49
9.4 Defining the use of HTTP headers.....	50
9.5 Configuring the Mail Sender Settings.....	51
9.6 Registering License Keys.....	53
9.7 Configuring the binary files repository.....	53
9.8 Working with database scripts.....	54
9.9 Defining the Domain Aliases.....	55
<b>10 Preferences.....</b>	<b>56</b>
10.1 Defining the default web service categories.....	56
10.2 Defining mapping tables preferences.....	57
10.3 Managing profiles and rights.....	58
10.3.1 Defining profiles.....	59

10.3.2 Assigning rights to profiles.....	59
10.3.3 Assigning profiles to users.....	59
10.4 Accessing your profile.....	59
10.5 Changing your password.....	60

## USER ADMINISTRATION

---

<b>Introduction to user administration.....</b>	<b>62</b>
<b>11 Users.....</b>	<b>63</b>
11.1 Creating users.....	63
11.2 Importing users.....	64
11.3 Editing users.....	64
11.3.1 Editing a users' ID.....	64
11.3.2 Defining a user's login credentials.....	65
11.3.3 Associating users with external LDAP login credentials.....	66
11.4 Assigning profiles to users.....	66
11.5 Deleting users.....	67
<b>12 User rights.....</b>	<b>68</b>

## CREATING APIS

---

<b>Introduction to creating APIs.....</b>	<b>70</b>
<b>13 IBM i connections.....</b>	<b>71</b>
13.1 Creating 5250 session configurations.....	71
13.2 Editing 5250 session configurations.....	71
13.3 Deleting 5250 session configurations.....	72
<b>14 Metadata.....</b>	<b>73</b>
14.1 Accessing the Metadata Designer.....	74
14.2 Creating metadata.....	75
14.2.1 Define screen IDs.....	76
14.2.2 Define field IDs.....	78
14.2.3 Define list IDs.....	79
14.2.4 Define column IDs.....	80
14.2.5 Define end of list IDs.....	81
14.2.6 Navigating through multi-page data.....	82
14.3 Duplicating metadata.....	83
14.4 Deleting metadata.....	84
<b>15 Scenarios.....</b>	<b>85</b>
15.1 Creating scenarios.....	86
15.2 Editing a scenario's ID.....	87
15.3 Recording scenarios.....	87
15.4 Viewing the steps in scenarios.....	88
15.5 Managing unexpected screen actions.....	89
15.6 Creating new scenarios based on existing steps.....	90
15.7 Testing scenarios.....	91

15.8 Deleting scenarios	91
<b>16 Web service descriptors</b>	<b>92</b>
16.1 Creating web service descriptors	93
16.2 Defining 5250 web service descriptors	94
16.2.1 Editing 5250 descriptor details	94
16.2.2 Editing the 5250 web service properties	95
16.2.3 Defining input and output content	95
16.2.4 Defining success messages	98
16.2.5 Defining a custom JSON output format for a web service	99
16.3 Defining SQL web service descriptors	103
16.3.1 Editing SQL descriptor details	103
16.3.2 Defining an SQL Select Query	104
16.3.3 The SQL execution process	105
16.4 Defining REST web service descriptors	106
16.4.1 Creating REST actions	107
16.4.2 Mapping a Variable Segment to an Input Parameter	107
16.5 Testing the execution of the web service	108
16.5.1 Execute a test of a web service	108
16.5.2 Execute a test of a web service step by step	109
16.5.3 View execution results	110
16.6 Exporting web service descriptors	110
16.7 Exporting web service descriptors' Open API specifications	112
16.8 Promoting descriptors to the production server	112
16.9 Promoting entire descriptor categories to the production server	113
16.10 Deleting web service descriptors	114

## HOSTING WEB SERVICES

---

<b>Introduction to hosting web services</b>	<b>116</b>
<b>17 IBM i Production server settings</b>	<b>117</b>
<b>18 Production packages</b>	<b>121</b>
18.1 Creating production packages	122
18.2 Editing production packages	123
18.3 Accessing version tags	123
<b>19 Version tags</b>	<b>124</b>
19.1 Accessing version tags	124
19.2 Creating version tags	125
19.3 Editing version tags	125
19.4 Promoting versions to the Kong server	126
19.5 Registering versions in APIGEE Edge	129
19.6 Exporting web services' Open API specifications	130
<b>20 Registered web services</b>	<b>131</b>
20.1 Importing web service descriptors	131
20.2 Demoting registered web services	132



<b>21 API Gateways settings</b> .....	<b>133</b>
21.1 APIGEE Gateway .....	133
21.1.1 APIGEE Settings .....	133
21.1.2 APIGEE Templates .....	134
21.1.2.1 Creating APIGEE API Proxy templates .....	135
21.1.2.2 Updating APIGEE API Proxy templates .....	135
21.1.2.3 Package variabilization Process .....	135
21.1.2.4 Substitution variables .....	136
21.2 WSO2 Settings .....	137
21.3 Kong Gateway .....	137

## APPENDICES

---

<b>Web services catalog</b> .....	<b>140</b>
<b>Glossary</b> .....	<b>143</b>

# Tables

---

Table 1: Contact ARCAD.....	3
Table 2: Related Documentation.....	4
Table 3: ARCAD-API User Guide Publication Record.....	4
Table 4: ARCAD-API Studio Hardware Requirements - Windows.....	20
Table 5: ARCAD-API Studio Software Requirements - Windows.....	20
Table 6: ARCAD-API Studio Hardware Requirements - Linux.....	21
Table 7: ARCAD-API Studio Software Requirements - Linux.....	21
Table 8: ARCAD-API Server Hardware Requirements - Windows.....	22
Table 9: ARCAD-API Server Software Requirements - Windows.....	22
Table 10: ARCAD-API Server Hardware Requirements - Linux.....	23
Table 11: ARCAD-API Server Software Requirements - Linux.....	23
Table 12: ARCAD-API Server Hardware Requirements - IBM i.....	24
Table 13: ARCAD-API Server Software Requirements - IBM i.....	24
Table 14: Input custom headers.....	50
Table 15: Output custom headers.....	50
Table 16: The user rights.....	68
Table 17: Reference tags to use in custom JSON output format.....	100
Table 18: Substitution Variable.....	136

# Figures

Figure 1: ARCAD-API in ARCAD's Modernization product suite .....	14
Figure 2: The ARCAD-API functional diagram .....	15
Figure 3: Access the elements in the designer server .....	16
Figure 4: Access the elements in the embedded production server .....	16
Figure 5: The ARCAD-API Studio navigator .....	18
Figure 6: The ARCAD-API Studio navigator .....	43
Figure 7: The Connection Properties dialog .....	44
Figure 8: The ARCAD-API Server connection splash screen .....	45
Figure 9: Configure the ARCAD-API Server .....	48
Figure 10: Configure the APIGEE general settings .....	48
Figure 11: Configure the API Gateway connection values .....	49
Figure 12: Configure the default location for descriptors .....	49
Figure 13: Configure the use of HTTP headers .....	51
Figure 14: Configure the mail sender settings .....	51
Figure 15: Register the ARCAD-API license key .....	53
Figure 16: The H2 database maintenance scripts .....	54
Figure 17: Manage the ARCAD-API Studio preferences .....	56
Figure 18: Define the default web service categories .....	56
Figure 19: Define the mapping tables .....	57
Figure 20: Define the mapping values .....	58
Figure 21: Defining Profiles and Rights .....	58
Figure 22: Current User Information (Preferences menu) .....	60
Figure 23: Change your password (Preferences menu) .....	60
Figure 24: User Management .....	63
Figure 25: The LDAP user management view .....	66
Figure 26: Accessing 5250 Session Configuration search view .....	71
Figure 27: Editing 5250 Session Configurations .....	72
Figure 28: The Metadata designer .....	74
Figure 29: Accessing the Metadata Designer .....	74
Figure 30: The overlapping field warning dialog .....	76
Figure 31: The captured Screen ID metadata .....	77
Figure 32: Tick the Is Password checkbox for identifying data fields .....	78
Figure 33: The captured Field ID metadata .....	79
Figure 34: The captured List ID metadata .....	80
Figure 35: The captured Column ID metadata .....	81
Figure 36: The captured End of List ID metadata .....	82
Figure 37: Deleting metadata .....	84
Figure 38: The Scenario editor .....	86
Figure 39: Start recording the scenario .....	88
Figure 40: Viewing the steps in a scenario .....	89
Figure 41: Adding unexpected screen actions to a step .....	90
Figure 42: The web service descriptor editor .....	93
Figure 43: Navigate through the screens of the scenario to define the input/output values .....	95
Figure 44: Drag a field ID into a descriptor to create input data .....	96
Figure 45: Drag a field ID into the Success Message ID field to create functional success cues .....	99
Figure 46: Web service test .....	108
Figure 47: Web service test .....	109
Figure 48: Contents of the export JSON file .....	111

Figure 49: The IBM i Production Server Settings view.....	117
Figure 50: Check how many days remain on your evaluation license.....	118
Figure 51: Production packages in the production server.....	122
Figure 52: The Version Tags search view.....	125
Figure 53: The Version Tag editor.....	126
Figure 54: The Registered Web Services view.....	131
Figure 55: Define the APIGEE settings.....	134
Figure 56: Define the WSO2 settings.....	137
Figure 57: Define the Kong settings.....	138
Figure 58: Web service documentation (5250 catalog).....	141
Figure 59: Web service documentation (SQL catalog).....	142



# ABOUT ARCAD-API

# 1 About ARCAD-API

Create and expose RESTful Web services over IBM i 5250 applications

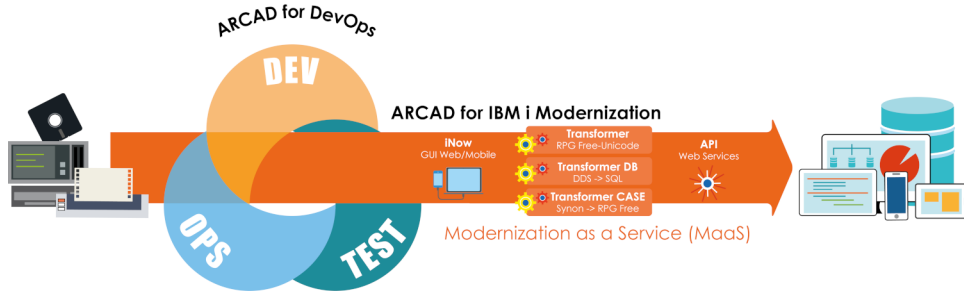


Figure 1: ARCAD-API in ARCAD's Modernization product suite

ARCAD API is a “no-code” solution for organizations needing to rapidly interface core business systems on IBM i with other applications and devices. This makes the solution particularly well adapted in situations with no access to application source code – such as vendor packages – or where the source code is too complex or too critical to modify.

ARCAD API enables you to create web services to interface directly with the native 5250 data stream. It is able to record 5250 scenarios, then to translate them into RESTful web services. It is also able to create web services based on an SQL query to extract data from the target IBM i database.

ARCAD API generate web services only with the working knowledge of how to use the IBM i application by recording scenarios of your applications' usage in the 5250 emulator of the API Studio. The web services are created and tested with API Studio and are deployed on an API Production Server or through API gateways.

ARCAD API is a client-server application. The studio provides access to both the design and production servers. Both servers are installed together. A production distribution of the ARCAD-API Server and the API Studio is also available only to manage the web services that are in production.

## Reference

For more information about installing ARCAD-API, refer to [Installation](#) on page 26.

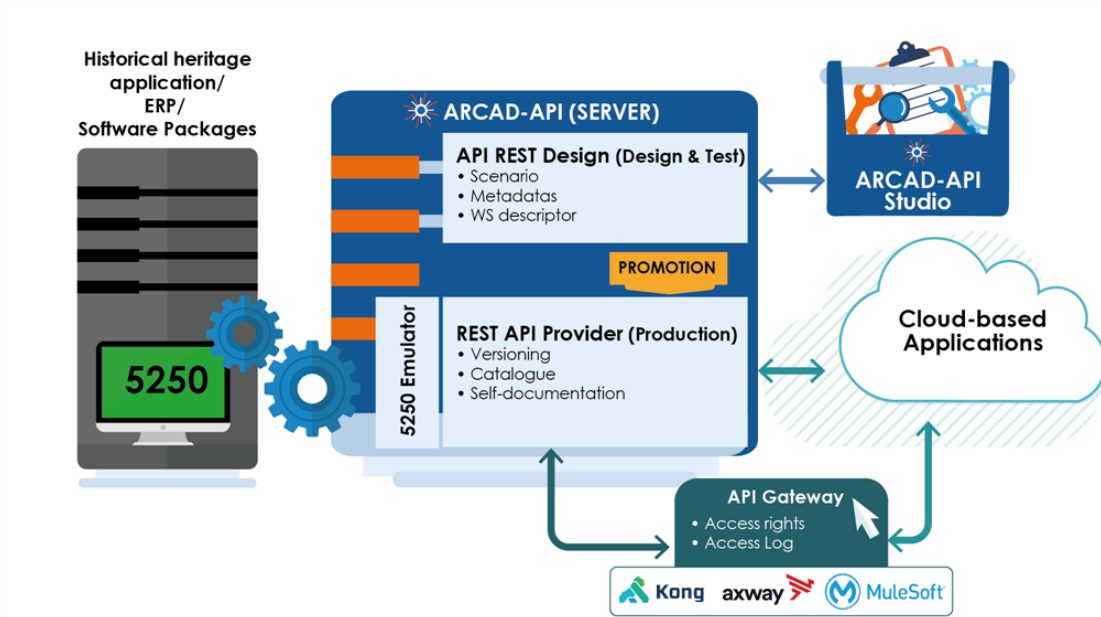


Figure 2: The ARCAD-API functional diagram

### The Studio

The ARCAD-API Studio is the client application which provides access to all of the elements required to create, test, validate and promote web services.

The studio connects to the ARCAD-API Server which has two types of servers embedded together.

**Reference**

For more information about using the ARCAD-API Studio, refer to the [Overview on page 18](#).

## The Design Server

The design server is the server where the metadata elements, the scenarios, and the web services are designed and tested. Once a web service is validated, it can be promoted into production.

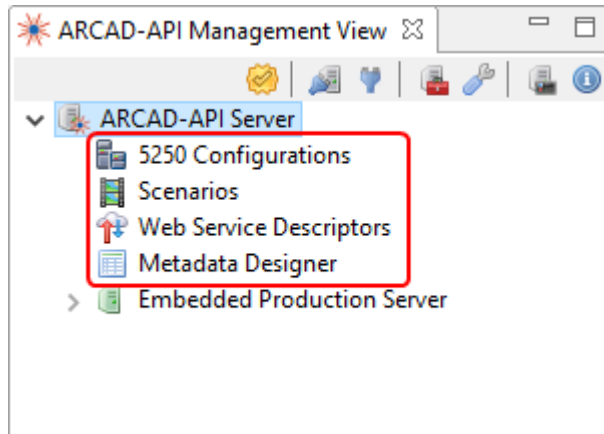


Figure 3: Access the elements in the designer server

### Reference

This server requires some configuration. For more information, refer to [Configuring ARCAD-API on page 41](#).

## The Production Server

The production server hosts web services in production. When the web services are validated they are promoted to production where they become available for general consumption. The production server allows you to manage versions, self-documentation, API packages and baselines and execution optimization.

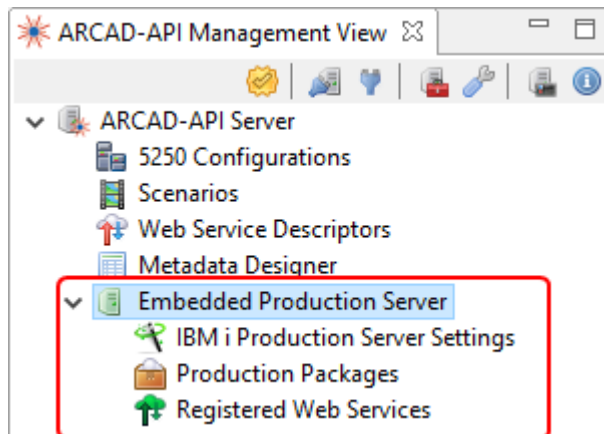


Figure 4: Access the elements in the embedded production server

### Reference

For more information about pushing web services to production, refer to [Hosting Web Services on page 115](#).



## 2 Main concepts

---

ARCAD API generate web services by recording scenarios of your applications' usage in the 5250 emulator of the API Studio and then translating them into web services. It also generates web services based on an SQL query to extract data from the target IBM i database. The web services are created and tested on the ARCAD-API designer server and are deployed on the API Production Server or through API gateways.

### On the design server:

1. Connect to the IBM i on which to record the scenarios.
2. Define all required **Metadata** information to be able to create and execute the service or procedure.
3. Record a **Scenario** of the service or procedure on the 5250 interface.
4. Create a **Web Service Descriptor**.
5. Test the web service.
6. Promote the web service to production.

### On the production server:

1. Create **Web Services Packages** and define versions.
2. Browse the promoted services and packages.
3. Refer to the documentation in the **web service catalog**.
4. Manage the different **API Gateways**.

## 3 Overview

---

The ARCAD-API Studio is an independent Eclipse rich client platform.

The design server and the embedded production server are both accessed from the ARCAD-API Studio which has only one perspective.

To verify the version of the ARCAD-API Studio, open the **Help** menu then select **About ARCAD-API Studio**. A window opens to display information about the installation details and the version.

### About perspectives

A perspective defines the initial set and layout of views in the workbench window. Within the window, each perspective shares the same set of editors. Each perspective provides a set of capabilities aimed at accomplishing a specific type of task or working with specific types of resources. For example, the JavaTM perspective combines views that you would commonly use while editing Java source files, while the Debug perspective contains views that you would use while debugging a program. Perspectives contain views and editors and control what appears in certain menus and tool bars.

Views in ARCAD-API are context sensitive. To see a view that is not currently active, open it by accessing the **Show View** dialog (*Window > Show view > Other*) and selecting the view. Re-position views by dragging and dropping them to the desired location in the studio. Click an edge and drag to re-size a view. Double-click on any view's toolbar to maximize it. This is helpful when a view contains a lot of information; maximizing the view makes it fill the entire window. Double-click again to minimize the view.

To re-set the perspective to the default layout, click *Window > Reset Perspective*.

### The navigator

This primary view provides access to all of the features available in the ARCAD-API product.

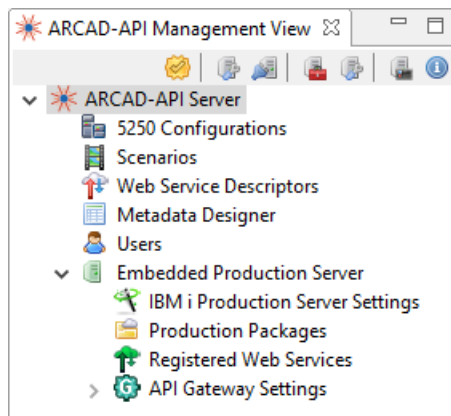


Figure 5: The ARCAD-API Studio navigator



# SYSTEM REQUIREMENTS

## 4 System requirements for the ARCAD-API Studio

### Chapter Summary

4.1 Windows.....	20
4.2 Linux.....	20

The ARCAD-API Studio can be installed on Windows or Linux.

### 4.1 Windows

#### 4.1.1 Hardware requirements

The following table lists the hardware components that are required for the ARCAD-API Studio on Windows x64.

Requirement	Value
System Architecture	64 bit processor
CPU	DUAL Core processor minimum
Physical memory (RAM)	2GB minimum, 4GB recommended
Virtual memory	<ul style="list-style-type: none"><li>• If physical memory is between 2GB and 16GB, then set virtual memory to 1x the size of the RAM</li><li>• If physical memory is more than 16GB, then set virtual memory to 16GB</li></ul>
Disk Space	Typical Install Type total: 1GB
Video adapter	256 colors
Screen Resolution	1024x768 minimum
Network	Any TCP/IP capable network interface

Table 4: ARCAD-API Studio Hardware Requirements - Windows

#### 4.1.2 Software requirements

The following table lists the software requirements for the ARCAD-API Studio on Windows x64.

Requirement	Value
Operating system	<ul style="list-style-type: none"><li>• Any Microsoft supported Windows OS 64bit</li></ul>
Network protocol	<ul style="list-style-type: none"><li>• TCP/IPv4</li><li>• TCP/IPv4 with SSL</li></ul>

Table 5: ARCAD-API Studio Software Requirements - Windows

### 4.2 Linux

## 4.2.1 Hardware requirements

The following table lists the hardware components that are required for the ARCAD-API Studio on Linux.

Requirement	Value
System Architecture	64 bit processor
Physical memory (RAM)	4GB minimum, 8GB recommended
Disk Space	Typical Install Type total: 1GB
Network	Any TCP/IP capable network interface

Table 6: ARCAD-API Studio Hardware Requirements - Linux

## 4.2.2 Software requirements

The following table lists the software requirements for the ARCAD-API Studio on Linux.

Requirement	Value
Operating system	<ul style="list-style-type: none"><li>Any Linux 64bit OS</li></ul>
Network protocol	<ul style="list-style-type: none"><li>TCP/IPv4</li><li>TCP/IPv4 with SSL</li></ul>

Table 7: ARCAD-API Studio Software Requirements - Linux

# 5 System requirements for the ARCAD-API Server

## Chapter Summary

5.1 Windows.....	22
5.2 Linux.....	23
5.3 IBM i.....	24

The ARCAD-API Server can be installed on Windows, Linux or IBM i.

## 5.1 Windows

### 5.1.1 Hardware requirements

The following table lists the hardware components that are required for the ARCAD-API Server on Windows x64.

Requirement	Value
System Architecture	64 bit processor
CPU	DUAL Core processor minimum
Physical memory (RAM)	2GB minimum, 4GB recommended <sup>1</sup>
Virtual memory	<ul style="list-style-type: none"> <li>If physical memory is between 2GB and 16GB, then set virtual memory to 1x the size of the RAM</li> <li>If physical memory is more than 16GB, then set virtual memory to 16GB</li> </ul>
Hard disk	RAID storage or any secured storage with redundancy, backup and monitoring
Disk Space	Typical Install Type total: 10GB <sup>2</sup>
Network	Any TCP/IP capable network interface

Table 8: ARCAD-API Server Hardware Requirements - Windows

### 5.1.2 Software requirements

The following table lists the software requirements for the ARCAD-API Server on Windows x64.

Requirement	Value
Operating system	<ul style="list-style-type: none"> <li>Any Microsoft supported server OS with Java (eg windows server 2016 any version)</li> </ul>

Table 9: ARCAD-API Server Software Requirements - Windows

<sup>1</sup>Physical memory is used by this entity and all subsequent processes, operating systems and any other software installed on the server. Please refer to additional software vendors for additional physical memory requirements.

<sup>2</sup>This size can be higher depending on the options selected and typical usage of the software, refer to Hard Disk Space requirement planning section to determine your needs.

Requirement	Value
Network protocol	<ul style="list-style-type: none"> <li>TCP/IPv4</li> <li>TCP/IPv4 with SSL</li> </ul>
Filesystem	NTFS or any journalized filesystem

Table 9: ARCAD-API Server Software Requirements - Windows

## 5.2 Linux

### 5.2.1 Hardware requirements

The following table lists the hardware components that are required for the ARCAD-API Server on Linux x64.

Requirement	Value
System Architecture	AMD64 and Intel EM64T
CPU	DUAL Core processor minimum
Physical memory (RAM)	2GB minimum, 4GB recommended <sup>1</sup>
Virtual memory	<ul style="list-style-type: none"> <li>If physical memory is between 2GB and 16GB, then set virtual memory to 1x the size of the RAM</li> <li>If physical memory is more than 16GB, then set virtual memory to 16GB</li> </ul>
Hard disk	RAID storage or any secured storage with redundancy, backup and monitoring
Disk Space	Typical Install Type total: 10GB <sup>2</sup>
Network	Any TCP/IP capable network interface

Table 10: ARCAD-API Server Hardware Requirements - Linux

### 5.2.2 Software requirements

The following table lists the software requirements for the ARCAD-API Server on Linux x64.

Requirement	Value
Operating system	Linux 64bits with support for Java 8 64 bits
Runlevel	Server should be started in multi-user mode, commonly runlevel 3 or 5
Network protocol	<ul style="list-style-type: none"> <li>TCP/IPv4</li> <li>TCP/IPv4 with SSL</li> </ul>

Table 11: ARCAD-API Server Software Requirements - Linux

<sup>1</sup>Physical memory is used by this entity and all subsequent processes, operating systems and any other software installed on the server. Please refer to additional software vendors for additional physical memory requirements.

<sup>2</sup>This size can be higher depending on the options selected and typical usage of the software, refer to Hard Disk Space requirement planning section to determine your needs.

Requirement	Value
Filesystem	Ext3, ext4 or any journalized filesystem
SELINUX	Disabled

Table 11: ARCAD-API Server Software Requirements - Linux

## 5.3 IBM i

### 5.3.1 Hardware requirements

The following table lists the hardware components that are required for the ARCAD-API Server on IBM i.

Requirement	Value
Physical memory (RAM)	RAM: 2GB minimum physical memory (4GB recommended) <sup>1</sup>
Disk Space	Typical Install Type total: 10GB <sup>2</sup>
Network	Any TCP/IP capable network interface

Table 12: ARCAD-API Server Hardware Requirements - IBM i

### 5.3.2 Software requirements

The following table lists the software requirements for the ARCAD-API Server on IBM i.

Requirement	Value
Operating system	IBM i $\geq 7.1$
Network protocol	<ul style="list-style-type: none"> <li>TCP/IPv4</li> <li>TCP/IPv4 with SSL</li> </ul>
Java JRE	JRE/JDK v8
At least one instance of the Remote Execution Server (REXEC) must be running.	

Table 13: ARCAD-API Server Software Requirements - IBM i

### 5.3.3 Host name resolution

Typically, the partition on which you want to install the ARCAD-API Server is connected to a network. Ensure that the partition host name is resolvable through a Domain Name System (DNS) or the centrally-maintained TCP/IP host table (go CFGTCP, option 10).

Use the PING command to ensure that your computer host name is resolvable.

<sup>1</sup>Physical memory is used by this entity and all subsequent processes, operating systems and any other software installed on the server. Please refer to additional software vendors for additional physical memory requirements.

<sup>2</sup>This size can be higher depending on the options selected and typical usage of the software, refer to Hard Disk Space requirement planning section to determine your needs.



 **Example**

```
PING RMTSYS(hostname)
Verifying connection to host system
hostname.domain.com at address 192.168.0.10.
PING reply 1 from 192.168.0.10 took 18 ms. 256 bytes. TTL 64.
```

If your computer host name does not resolve, contact your system administrator.

### 5.3.4 Users for the ARCAD-API Server

Log into the system with a profile with the following special authorities:

- \*ALLOBJ
- \*JOBCTL

 **Important!**

These rights are required only when installing the product.



# INSTALLATION

# 6 Installing the ARCAD-API Studio

## Chapter Summary

6.1 Prerequisites.....	27
6.2 Windows.....	27
6.3 Linux.....	29

The ARCAD-API Studio can be installed on Windows or Linux.

## 6.1 Prerequisites

### Reference

For complete details about the technical prerequisites for ARCAD-API, refer to the [System requirements on page 19](#).

## 6.2 Windows

The installation process takes approximately 5 minutes. It is a simple procedure similar to most other Windows applications. For trouble-free installation, it is recommended that you close all active Windows applications before beginning the installation.

### Note

If the tool is already installed on your computer, reinstalling it will update it to the new version automatically.

### 6.2.1 Install

Follow the subsequent steps to install the ARCAD-API Studio on Windows.

**Step 1** Copy the .exe file to your machine and execute it to launch the wizard. Your profile must have administrator privileges to run the execution file.

**Step 2** From the **Select Setup Language** window, select the language for the installation.

**Step 3** Review and accept the license agreement.

The **License Agreement** page presents the ARCAD license agreement for you to review. Please read it carefully. When you have reviewed the agreement, select **I accept the agreement**.

**Step 4** Confirm the installation location.

The **Select Destination Location** screen displays the default location where ARCAD elements will be installed. If you prefer to install the software elements in a different location, either type in the location, or click **Browse...** to navigate to and select the alternate location.

The default root location is `C:\Program files\ARCAD Solutions\`.

A sub-folder for each ARCAD application you install will be created at this location.

**Warning!**

The access rights to this folder can then be restricted to the user used to run the Server's Windows service.

**Step 5** Select the tool(s) to install. To install only the ARCAD-API Studio, unselect the server.

**Reference**

To install both tools, leave both options selected then refer to [Installing the ARCAD-API Server on page 31](#) for instructions on how to manage the ARCAD-API Server options.

**Step 6** Confirm the program's shortcut location.

The **Select Start Menu Folder** screen displays the default location where a shortcut will be created in your computer's Start menu. If you prefer to install the shortcut under a different group, either type the location here, or click **Browse...** to navigate to and select the alternate location.

The default root folder is *ARCAD Solutions\*.

If you do not want to create a start menu folder, select the **Don't create a Start Menu folder** checkbox.

**Step 7** Install.

The **Ready to Install** screen enables you to review and change or confirm the setup parameters provided and to launch the installation.

To change a setup parameter, click **< Back** and return to the necessary screen.

If you agree with the installation parameters displayed, click **Install** to start the copy phase of the installation process. The process may take a moment while the windows services are launched.

As the installation proceeds, a status bar displays its progress. Each element copied appears above the status bar. Click **Cancel** to interrupt the installation.

**Step 8** Complete the setup process. The final page of the wizard displays confirmation that the setup was a success. Click **Finish** to close the setup wizard.

Check the **Launch** check box to automatically open the ARCAD-API Studio.

**Result** The ARCAD-API Studio is installed and available for use.

## 6.2.2 Update

Updating the ARCAD-API Studio is similar to the installation process in that it is launched via the installation .exe.

**Note**

If you are updating without uninstalling, the majority of the user-defined configurations in the studio are saved in the user directory (the default is: *C:\Users\<user>\arcad\<module>*).

Follow the subsequent steps to update the ARCAD-API Studio on Windows.

**Step 1** Copy the `.exe` file to your machine and execute it to launch the wizard. Your profile must have administrator privileges to run the execution file.

**Step 2** From the **Select Setup Language** window, select the language for the installation.

**Step 3** Review and accept the license agreement.

The **License Agreement** page presents the ARCAD license agreement for you to review. Please read it carefully. When you have reviewed the agreement, select **I accept the agreement**.

**Step 4** Install.

The **Ready to Install** screen enables you to review and change or confirm the setup parameters provided and to launch the installation.

To change a setup parameter, click **< Back** and return to the necessary screen.

If you agree with the installation parameters displayed, click **Install** to start the copy phase of the installation process. The process may take a moment while the windows services are launched.

As the installation proceeds, a status bar displays its progress. Each element copied appears above the status bar. Click **Cancel** to interrupt the installation.

**Step 5** Complete the setup process. The final page of the wizard displays confirmation that the setup was a success. Click **Finish** to close the setup wizard.

Check the **Launch** check box to automatically open the ARCAD-API Studio.

**Result** The ARCAD-API Studio is updated and available for use.

## 6.2.3 Uninstall

Follow the subsequent steps to uninstall the ARCAD-API Studio on Windows.

**Step 1** Log into the system with an administrator account.

**Step 2** Either launch the `uninstall.exe` located in the installation directory

- or -

Open **Add or Remove Programs** (*Start > Control Panel*), find the module in the list of installed software and select **Uninstall**.

**Step 3** Remove any remaining files in the installation path and remove any remaining configuration files in the user directory.

**Result** The ARCAD-API Studio is completely uninstalled.

## 6.3 Linux

### 6.3.1 Install

Follow the subsequent steps to install the ARCAD-API Studio on Linux.

**Step 1** Transfer the `.tar.gz` to the `/tmp` directory on the target system.

**Step 2** Open a terminal session and go to the installation directory: `cd /opt`

**Step 3** Execute the following command to extract the `tar.gz` archive into the current directory:

```
tar xzf API-Studio.tar.gz
```

**Step 4** Change the ownership of the installation directory to give it to the user and group that will run the ARCAD-API Studio:

```
chown username:groupname -R /opt/API-Studio
```

**Result** The ARCAD-API Studio is installed and available for use.

### 6.3.2 Update

---

Follow the subsequent steps to update the ARCAD-API Studio on Linux.

**Step 1** Completely delete the API-Studio directory:

```
rm -rf /opt/API-Studio
```

**Step 2** Install the ARCAD-API Studio from scratch following the installation instructions.

**Result** The ARCAD-API Studio is updated and available for use.

### 6.3.3 Uninstall

---

To uninstall the ARCAD-API Studio on Linux, remove the software from the installation directory.

```
rm -rf /opt/API-Studio
```

The ARCAD-API Studio is completely uninstalled.

# 7 Installing the ARCAD-API Server

---

## Chapter Summary

7.1 Prerequisites.....	31
7.2 Windows.....	31
7.3 Linux.....	34
7.4 IBM i.....	37

The ARCAD-API Server centralizes all the necessary elements to create and maintain APIs. The server is the core of ARCAD-API because it is required to use the ARCAD-API Studio. This server imbeds both the designer and production servers.

The ARCAD-API Server can be installed on Windows, Linux or IBM i.

## 7.1 Prerequisites

---

### Reference

For complete details about the technical prerequisites for ARCAD-API, refer to the [System requirements on page 19](#).

## 7.2 Windows

---

The ARCAD-API Server runs as a Windows service.

The installation process takes approximately 5 minutes. It is a simple procedure similar to most other Windows applications. For trouble-free installation, it is recommended that you close all active Windows applications before beginning the installation.

### Note

If the tool is already installed on your computer, reinstalling it will update it to the new version automatically.

### 7.2.1 Install

---

Follow the subsequent steps to install the ARCAD-API Server on Windows.

**Step 1** Copy the .exe file to your machine and execute it to launch the wizard. Your profile must have administrator privileges to run the execution file.

**Step 2** From the **Select Setup Language** window, select the language for the installation.

**Step 3** Review and accept the license agreement.

The **License Agreement** page presents the ARCAD license agreement for you to review. Please read it carefully. When you have reviewed the agreement, select **I accept the agreement**.

**Step 4** Confirm the installation location.

The **Select Destination Location** screen displays the default location where ARCAD elements will be installed. If you prefer to install the software elements in a different location, either type in the location, or click **Browse...** to navigate to and select the alternate location.

The default root location is `C:\Program files\ARCAD Solutions\`.

A sub-folder for each ARCAD application you install will be created at this location.

 **Warning!**

The access rights to this folder can then be restricted to the user used to run the Server's Windows service.

**Step 5** Install.

The **Ready to Install** screen enables you to review and change or confirm the setup parameters provided and to launch the installation.

To change a setup parameter, click **< Back** and return to the necessary screen.

If you agree with the installation parameters displayed, click **Install** to start the copy phase of the installation process. The process may take a moment while the windows services are launched.

As the installation proceeds, a status bar displays its progress. Each element copied appears above the status bar. Click **Cancel** to interrupt the installation.

**Step 6** Define the TCP ports to use.

 **Important!**

The ports defined here will be used to configure the connection.

Press **Enter/Next** to use the default ports.

The ARCAD-API Server is an independent AFS server with an embedded H2 database.

The API Server's default HTTP port is 5260, and the default HTTPS port is 52600.

To disable a port, enter 0 (zero) when asked for the port number.

 **Important!**

If you are using other ARCAD Group products on the same host, it is recommended to use the default ports to avoid any conflicts with other servers.

If an application is already using it, the default port number is incremented until a free port number is found. It is possible to manually change the port after installation.

**Step 7** Complete the setup process. The final page of the wizard displays confirmation that the setup was a success. Click **Finish** to close the setup wizard.

**Step 8** Verify the Windows Service is running.

The final screen of the installation process enables you to automatically open the Local Services Management Console. If the **Open Windows service** checkbox is selected, the **Services** window opens after clicking **Finish**. This window enables you to verify that its status is set to "Started".

**Result** The ARCAD-API Server is installed and available for use.



## 7.2.2 Update

Updating the ARCAD-API Server is similar to the installation process in that it is launched via the installation .exe. However, after the update is complete, the database must also be updated.

### Note

If you are updating without uninstalling, the majority of the installation preferences are saved and automatically reused.

To change the saved values, uninstall completely and define new values by reinstalling from scratch.

Follow the subsequent steps to update the ARCAD-API Server on Windows.

**Step 1** Copy the .exe file to your machine and execute it to launch the wizard. Your profile must have administrator privileges to run the execution file.

**Step 2** From the **Select Setup Language** window, select the language for the installation.

**Step 3** Review and accept the license agreement.

The **License Agreement** page presents the ARCAD license agreement for you to review. Please read it carefully. When you have reviewed the agreement, select **I accept the agreement**.

**Step 4** Install.

The **Ready to Install** screen enables you to review and change or confirm the setup parameters provided and to launch the installation.

To change a setup parameter, click **< Back** and return to the necessary screen.

If you agree with the installation parameters displayed, click **Install** to start the copy phase of the installation process. The process may take a moment while the windows services are launched.

As the installation proceeds, a status bar displays its progress. Each element copied appears above the status bar. Click **Cancel** to interrupt the installation.

**Step 5** Complete the setup process. The final page of the wizard displays confirmation that the setup was a success. Click **Finish** to close the setup wizard.

**Step 6** Verify the Windows Service is running.

The final screen of the installation process enables you to automatically open the Local Services Management Console. If the **Open Windows service** checkbox is selected, the **Services** window opens after clicking **Finish**. This window enables you to verify that its status is set to "Started".

**Result** The ARCAD-API Server is updated and available for use.

After any server update, you need to update the server's integrated database, to ensure that the database is fully compatible with the updated server.

### Important!

This step must be done **before starting the server service**.

To update the database, run the database update script supplied with the installation. The **dbupdate** file located in the **.\tools** folder.

## 7.2.3 Uninstall

Follow the subsequent steps to uninstall the ARCAD-API Server from Windows.

**Step 1** Log into the system with an administrator account.

**Step 2** Either launch the *uninstall.exe* located in the installation directory

- or -

Open **Add or Remove Programs** (*Start > Control Panel*), find the module in the list of installed software and select **Uninstall**.

**Step 3** Remove any remaining files in the installation path and remove any remaining configuration files in the user directory.

**Result** The ARCAD-API Server is completely uninstalled.

## 7.3 Linux


### 7.3.1 Install

Follow the subsequent steps to install the ARCAD-API Server on Linux.

**Step 1** Transfer the *.tar.gz* to the */tmp* directory on the target system.

**Step 2** Open a terminal session and go to the installation directory.

 **Example**  
cd /opt

 **Note**  
By default, the server is intended to be installed in the */opt* directory. If you install it in a different directory, you will have to update the server's scripts (see [Step 7 Edit the scripts.](#)).

**Step 3** Execute the following command to extract the *tar.gz* archive into the current directory:


```
tar xzf API-Server-[version number].tar.gz
```

**Step 4** [*Optional*] If you need to change the listening port(s) for the server, you must do so now.

The ARCAD-API Server is an independent AFS server with an embedded H2 database.

The API Server's default HTTP port is 5260, and the default HTTPS port is 52600.

To disable a port, enter 0 (zero) when asked for the port number.

 **Important!**  
If you are using other ARCAD Group products on the same host, it is recommended to use the default ports to avoid any conflicts with other servers.

To change the ports, open the `configuration/com.arcadsoftware.server.restful.cfg` in a text editor and change the following properties:

- **port**=<new HTTP port number> -or- <0> (zero) to disable the HTTP port
- **portssl**=<new HTTPS port number> -or- <0> (zero) to disable the HTTPS port

**Step 5** Create a specific user and group to run the server's process with the command:

```
adduser --system --no-create-home --group name
```

**Step 6** Change the ownership of the installation directory to give it to the user and group that will run the ARCAD-API Server:

```
chown username:groupname -R /opt/API-Server
```

**Step 7** Edit the scripts.

Open the following scripts in a text editor to set the variables to match your execution environment:

- `bin/API-Server`
  - Line 8: set **AFS\_HOME** to the installation directory if it was not installed in `/opt/...` by default.
  - Line 12: set **AFS\_USER** to the new user.
- `bin/API-Server.service`
  - Lines 8-14: update the paths so they match the installation directory.
  - Line 16: set the user to the new user.
  - Line 17: set the group to be the new user's or any group of your choice.
- configuration files in the `configuration` folder

**Step 8** [Optional] Execute the following commands to install the ARCAD-API Server as a systemd service:

```
cd /etc/systemd/system
systemctl link /opt/API-Server/bin/API-Server.service
systemctl enable API-Server
systemctl start API-Server
```

**Result** The ARCAD-API Server is installed and available for use.

#### Note

The ARCAD-API Server embeds, in the Linux packaging, the runtime of an open-source version of Java (OpenJDK 1.8). Depending on the needs of the host machine, this runtime can be replaced by an equivalent version managed by the operating system. To do so, delete the **jre** folder located in the server installation folder.

## 7.3.2 Update

After the ARCAD-API Server is updated, the database must also be updated.

Follow the subsequent steps to update the ARCAD-API Server on Linux.

**Step 1** Transfer the updated `API-Server.tar.gz` and the `API-Server-[version number]_Update.sh` script to the `/tmp` directory on the target system.

**Step 2** Open a terminal session and go into the `/tmp` directory: `cd /tmp`

**Step 3** Mark the `_Update.sh` as being executable:

```
chmod +x API-Server-[version number]_Update.sh
```

**Step 4** Open the `_Update.sh` with a text editor and make the following changes:

- Line 9: uncomment and set the **AFS\_HOME** variable to the installation directory.
- Line 10: change the **AFS\_URL** to match the address and port the server is listening on.
- Line 12: change the **AFS\_OSUSER** to the user running the server.
- Line 13: change the **AFS\_OSGROUP** to the group running the server.
- Lines 16-19: uncomment the line corresponding to the way the server must be started.
- Lines 22-25: uncomment the line corresponding to the way the server must be stopped.

**Step 5** Run the `_Update.sh`:

```
./API-Server-[version number]_Update.sh
```

**Note**

You can keep the `_Update.sh` for future updates, unless stated otherwise in the release notes.

**Result** The ARCAD-API Server is updated and available for use.

After any server update, you need to update the server's integrated database, to ensure that the database is fully compatible with the updated server.

**Important!**

This step must be done **before starting the server service**.

To update the database, run the database update script supplied with the installation. The **dbupdate** file located in the `.\tools` folder.

### 7.3.3 Uninstall

Follow the subsequent steps to uninstall the ARCAD-API Server on Linux.

**Step 1** Stop the service using the service manager of your Linux distribution or use the script provided.



**Example**

Service command:

```
systemctl stop API-Server
```

Script command:

```
/opt/API-Server/bin/API-Server stop
```

**Step 2** Uninstall the service, if any, using the tools from your Linux distribution.

```
Example
systemctl disable API-Server
rm /etc/systemd/system/API-Server.service
systemctl daemon-reload
systemctl reset-failed
```

**Step 3** Remove the software from the installation directory.

```
Example
rm -rf /opt/API-Server
```

**Step 4** Remove any remaining files based on the server's configuration.

**Result** The ARCAD-API Server is completely uninstalled.

## 7.4 IBM i

The ARCAD-API Server can be installed and updated manually on IBM i or, depending on your security policy, remotely from any machine running Java.

**Note**  
If the tool is already installed on your computer, reinstalling it will update it to the new version automatically.

### 7.4.1 Install

Follow the subsequent steps to install the ARCAD-API Server on IBM i.

**Step 1** Copy the installation *.jar* file to any directory (such as the */tmp*) on the target IBM i IFS or the machine that will orchestrate the remote installation.

**Step 2** Launch the installation setup.

If you are installing manually on IBM i:

1. Open a session on the target IBM i using the QSECOFR profile or an equivalent.
2. Open a command line interpreter using the command QSH. Reach the location where you have copied your installation file.
3. In the command line interpreter, launch the setup using the installation command.

```
Example
java -jar Setup_API-Server-x.x.x._IBM i.jar
```

If you are installing remotely from Windows or Linux:

1. Launch the setup from the location on the machine orchestrating the installation using the command above. This will open a prompt asking for the **Remote IBM i name/address**.
2. Enter the name/address of the target IBM i.
3. Login using the QSECOFR profile or an equivalent.


**Step 3** Define the **AFS Starter Installation library** and iASP which contain the AFS Starter (a utility program used to start the ARCAD-API Server on IBM i).

By default they are **AFSSTARTER** and **\*SYSBAS**.

**Step 4** Define the installation location.

 **Example**  
/HOME/...

**Step 5** Define the TCP ports to use.


 **Important!**  
The ports defined here will be used to configure the connection.

Press **Enter/Next** to use the default ports.

The ARCAD-API Server is an independent AFS server with an embedded H2 database.

The API Server's default HTTP port is 5260, and the default HTTPS port is 52600.


To disable a port, enter 0 (zero) when asked for the port number.

 **Important!**  
If you are using other ARCAD Group products on the same host, it is recommended to use the default ports to avoid any conflicts with other servers.

If an application is already using it, the default port number is incremented until a free port number is found. It is possible to manually change the port after installation.

**Step 6** Define the **Job user** that will run the ARCAD-API Server.

**Step 7** Define the job queue library (**\*LIBL**) then the job queue (**JOBQ**) in which the job will be submitted.

 **Example**  
Job queue library [ARCAD\_SYS]  
Job queue [ARCAD\_CTL]


**Step 8** Define the name of the AFS instance to register in the AFSSTARTER.


By default it is **PRODUCT-AFS-ID**.

**Result** The ARCAD-API Server is installed and available for use.

## 7.4.2 Update

Updating the ARCAD-API Server is similar to the installation process in that it is launched via the installation *jar* file. However, after the update is complete, the database must also be updated.

 **Note**  
If you are updating without uninstalling, the majority of the installation preferences are saved and automatically reused.

 To change the saved values, uninstall completely and define new values by reinstalling from scratch.


Follow the subsequent steps to update the ARCAD-API Server on IBM i.

**Step 1** Copy the installation *.jar* file to any directory (such as the */tmp*) on the target IBM i IFS or the machine that will orchestrate the remote installation.

**Step 2** Launch the installation setup.

If you are installing manually on IBM i:

1. Open a session on the target IBM i using the QSECOFR profile or an equivalent.
2. Open a command line interpreter using the command QSH. Reach the location where you have copied your installation file.
3. In the command line interpreter, launch the setup using the installation command.

 **Example**  
`java -jar Setup_API-Server-x.x.x._IBM i.jar`

If you are installing remotely from Windows or Linux:

1. Launch the setup from the location on the machine orchestrating the installation using the command above. This will open a prompt asking for the **Remote IBM i name/address**.
2. Enter the name/address of the target IBM i.
3. Login using the QSECOFR profile or an equivalent.


**Step 3** Define the installation location.

 **Example**  
`/HOME/...`

Point to original installation location to use the same settings defined for the original installation (ports, installation location, service user).

**Result** The ARCAD-API Server is updated and available for use.

After any server update, you need to update the server's integrated database, to ensure that the database is fully compatible with the updated server.

 **Important!**  
 This step must be done **before starting the server service**.

To update the database, run the database update script supplied with the installation. The **dbupdate** file located in the `.\tools` folder.

### 7.4.3 Uninstall

Follow the subsequent steps to uninstall the ARCAD-API Server on IBM i.

**Step 1** Open a session on the IBM i where the server is installed, using the QSECOFR profile or an equivalent.

**Step 2** Stop and delete the service using the following commands:

```
ADDLIBLE <AFSSTARTER LIBRARY NAME>  
ENDAFSSVR APISERVER  
DLTAFSSVR APISERVER DELETE(*YES)
```

**Result** The ARCAD-API Server is completely uninstalled.





# CONFIGURING ARCAD-API

# Introduction to ARCAD-API configuration

---

This section describes how to configure the ARCAD-API Studio and the ARCAD-API Server.

It is not required to modify the default configuration settings. Configuration and preference options are specific to a studio and accessible at any time.

**Note**

Configuration and preference options are not specific to a user account. The configuration options are kept the same no matter which user account is used to access the workstation.

## 8 Managing the ARCAD-API Server

### Chapter Summary

8.1 Creating an ARCAD-API Server connection.....	43
8.2 Connecting to the ARCAD-API Server.....	44
8.3 Configuring the TLS settings.....	45
8.4 Configuring the ARCAD-API Server.....	46
8.5 Managing the ARCAD-API Server preferences.....	46
8.6 Viewing ARCAD-API Server logs.....	46
8.7 Verifying ARCAD-API Server versions.....	47

The ARCAD-API Studio must connect to the ARCAD-API Server to access and store all the elements required to use ARCAD-API.

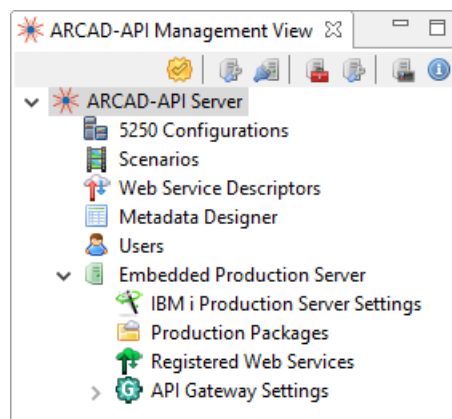


Figure 6: The ARCAD-API Studio navigator

### 8.1 Creating an ARCAD-API Server connection

The ARCAD-API Studio is set up to automatically connect to the server installed at this location: `http://localhost:5260`. This is because this is the default location/port in which the server is installed if both the server and the studio are installed at the same time on the same machine.

If the server is installed in the default location, there are no connection parameters to define. You can connect to the server directly.

#### Reference

[Connecting to the ARCAD-API Server on the next page](#)

Follow the subsequent steps to connect to a *different* ARCAD-API Server if you did not install the server in the default location.

**Step 1** Click the  server connection icon in the  navigator to open the **Connection Properties** dialog.

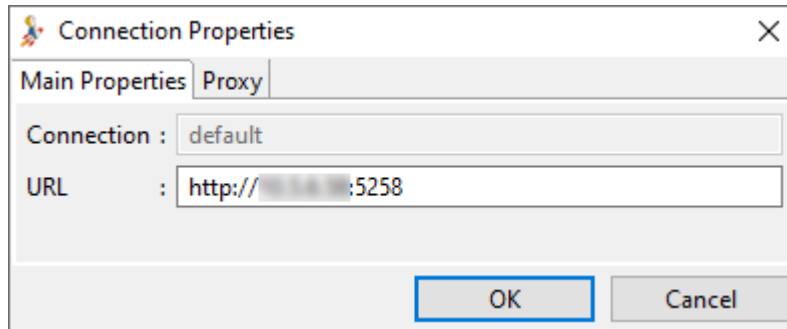


Figure 7: The Connection Properties dialog

**Step 2** Define the following fields in the **Connection Properties** dialog. Click **OK**.

#### Connection

This field displays the default name of the ARCAD-API Server connection.

#### Address/IP & Ports

Define the address where the ARCAD-API Server is located, including its appropriate port. The default HTTP port is 5260, and for secure connection, the default HTTPS port is 52600.

#### Proxy (tab)

[*Optional*] When this box is checked, an HTTP proxy is used to connect to the ARCAD-API Server. All the **Proxy** connection fields must be defined.

## 8.2 Connecting to the ARCAD-API Server

---

Follow the subsequent steps to connect the ARCAD-API Studio to the ARCAD-API Server.

**Step 1** Click the  connection icon in the  navigator to connect to the ARCAD-API Server.

**Step 2** Define the **Username** and **Password**.

Initially, the administration-level login must be used. This default generated admin-level user has access to all of the ARCAD-API features.

<b>Login</b>	admin@quadra
<b>Password</b>	quadra


Click **Connect**.



Figure 8: The ARCAD-API Server connection splash screen

**Result** The ARCAD-API Studio is connected to the ARCAD-API Server and is ready to be used.

**Note**

To change users, click the  Connection icon in the navigator. Connect to the ARCAD-API Server using a different **Username** and **Password**.

## 8.3 Configuring the TLS settings

**Important!**

This section does not describe the configuration required for TLS in the ARCAD-API Server. The TLS configuration belongs to the AFS configuration layer. Refer to the ARCAD-API Server Configuration Guide.

For security reasons, you may need to use an encrypted connection between the server and the studio. To do this you must activate the use of the HTTPS protocol, by checking the **Use HTTPS** option in the server connection parameters. However, this protocol, based on the TLS network protection layer, must be correctly configured to be implemented. Depending on the server settings, the client program will have to validate the server's TLS certificate and potentially use its own key.

In most cases, only the validation of the server certificate is required. If this certificate is the product of a chain of trust whose root is a recognized Certification Authority (CA) then there is nothing special to do on the client side. The Java virtual machine already recognizes and approves certificates from all the parties in the market. However, if the server's certificate was not issued by a CA, or your organization uses a specific CA, it is necessary to add the certificate of this CA, or directly the server's certificate in the studio configuration.

It is also possible that the server has been configured in such a way that the studio must also use a key pair (private, public) that requires the server to validate the studio's certificate.

In case the server uses a certificate that is not issued by a CA or requires the studio to use a key pair, the administrator will pass these materials to you. The most secure way to do this is to send them to you in a container, called a KeyStore, which is password protected. In case the HTTPS connection requires both the validation of the server's certificate and a key pair for the studio, it is possible that the administrator will send you two stores, one containing the certificate to be approved, this is called a TrustStore, the other containing only the studio's key pair. This last store keeps the name KeyStore.

It is also possible that the administrator transmits the certificate directly to you, without integrating it into a store, in this case make sure that the source of the transmission is not spoofed. Integrating the wrong trusted certificate can jeopardize the security of the entire installation.

 **Note**

Certificates and key pairs have a limited validity period. This period varies, generally from 1 to 12 months. The following configuration procedure must therefore be repeated each time the equipment is updated. Knowing that an expired certificate or key pair will block the connection to the server.

Once you have the KeyStore(s) and their respective passwords, or, if not available, the server certificate, follow the steps below to redefine the HTTPS protocol settings:

**Step 1** To access the **TLS Settings** dialog, click on the  **TLS Settings** icon in the navigator.

**Step 2** Edit the absolute paths to the **TrustStore** and **KeyStore** and their corresponding passwords.

**Step 3** Click the **Import Certificate** button to import your own certificates into the file indicated in the **TrustStore** path.

You can click the **Import Certificate** button to import the server's certificate if it was sent to you as is. It will be included in the TrustStore defined in this window.

**Step 4** Click **OK** to save your changes or **Cancel** to keep the default settings.

 **Note**

The **Reset** button resets access information and passwords to the default values provided by the ARCAD-API Studio.

## 8.4 Configuring the ARCAD-API Server

---

To configure the ARCAD-API Server, click on the  configure icon in the navigator.

 **Reference**

For more information, refer to [Configuring the ARCAD-API Server on page 48](#).

## 8.5 Managing the ARCAD-API Server preferences

---

To manage the ARCAD-API Server preferences, click on the  preferences icon in the navigator.

 **Reference**

For more information, refer to [Preferences on page 56](#).


## 8.6 Viewing ARCAD-API Server logs

---

To view a server's log, right-click on it in the  **Connections** list then select  **Server Log**.

## 8.7 Verifying ARCAD-API Server versions

---

To verify the version of an ARCAD-API Server, click on the  Server Information icon in the navigator. A dialog displays the selected server's version and the user with which you are currently connected.

## 9 Configuring the ARCAD-API Server

To configure the ARCAD-API Server, click on the  configure icon in the navigator.

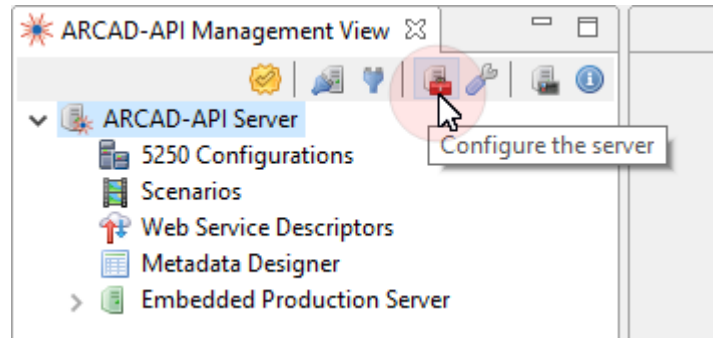


Figure 9: Configure the ARCAD-API Server

To open the configuration settings from a navigator, enter the following URL:  
`<address>:<port>/console.`

### **Warning!**

Enabling the H2 console on production environment is not recommended, it may expose the server to security vulnerabilities, especially if the connections are not limited to the localhost. It may also expose other H2 Database servers hosted on other server, if they also accept remote connections.

### 9.1 Defining the APIGEE Edge Settings

Access  ARCAD-API APIGEE Edge Settings →  APIGEE Edge General Settings

Change the default location in which the APIGEE templates temporarily stored.

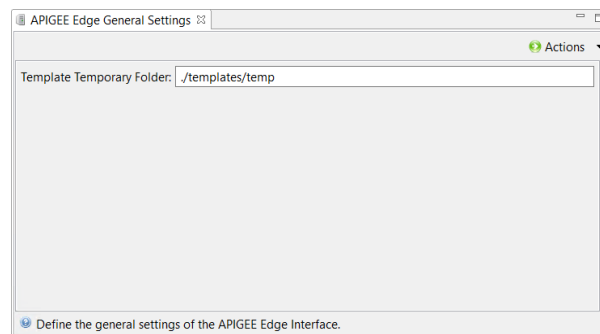



Figure 10: Configure the APIGEE general settings

### **Important!**

Ctrl+s will not save your changes in the server configuration editors. To save, click the  **Actions** drop-down menu and select **Save**.

### **Reference**

For more information about APIGEE Templates, refer to [APIGEE Templates on page 134](#).



## 9.2 Defining the API Gateway General Settings

Access  ARCAD-API Common Settings →  API Gateway General Settings

Change the default values for the API Gateways to connect to the ARCAD-API Server.

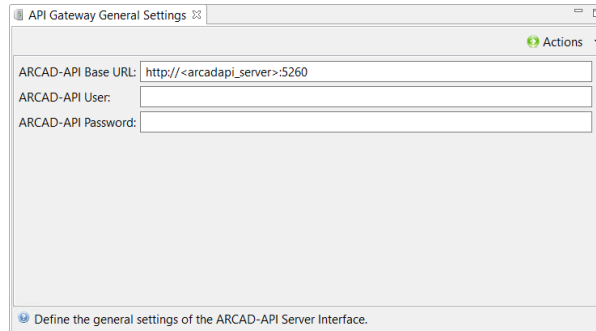


Figure 11: Configure the API Gateway connection values


### ARCAD-API Base URL

The base URL of the ARCAD-API Server. This URL will be used to define a service in the API Gateways.

### ARCAD-API User & Password

A valid ARCAD-API user and password. This user will be used by the API gateways to execute all the reachable web services on the ARCAD-API Server.

#### Important!

Ctrl+s will not save your changes in the server configuration editors. To save, click the  **Actions** drop-down menu and select **Save**.

#### Reference

For more information about API Gateways, refer to [API Gateways settings on page 133](#).

## 9.3 Defining the Web Service Descriptors Storage Area

Access  ARCAD-API Descriptor Settings →  General Settings

Change the default location in which the web descriptors created using the ARCAD-API Studio are saved. The default location is `./descriptors` on the ARCAD-API Server.

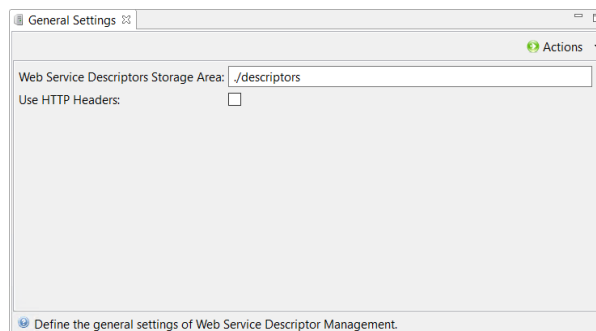



Figure 12: Configure the default location for descriptors

**Important!**  
 Ctrl+s will not save your changes in the server configuration editors. To save, click the  **Actions** drop-down menu and select **Save**.

**Reference**  
 For more information about web descriptors, refer to [Web service descriptors on page 92](#).

## 9.4 Defining the use of HTTP headers

Access	 ARCAD-API Descriptor Settings →  General Settings  ARCAD-API Design Execution Settings →  Execution Settings
--------	--

This option is available for both the API Server ("Design" server) and the API Production Server.

To simplify the call and the analysis of the JSON result of a web service, some information can be passed or retrieved using custom headers.

### Input Custom Headers

These headers can be used to pass some technical information when calling a web service:

Custom Header	Parameter	Description	Ws Type	Scope
arcapi-ibmi-host	server.host	The IP Address or DNS of the IBM i server	ALL	Design
arcapi-ibmi-ccsid	server.ccsid	The CCSID used by the connection	ALL	Design / Production
arcapi-ibmi-session	server.session	The 5252-session name	ALL	Design / Production
arcapi-ibmi-login	server.user	The IBMi User login	SQL	Design / Production
arcapi-ibmi-password	server.password	The IBMi Password	SQL	Design / Production

Table 14: Input custom headers

### Output Custom Headers

These headers can be used to retrieve some technical values

Custom Header	Description	Scope
arcapi-x-status	Functional Status	Design
arcapi-x-result	Execution Result	Design / Production
arcapi-x-instance	Execution Instance ID.	Design / Production

Table 15: Output custom headers

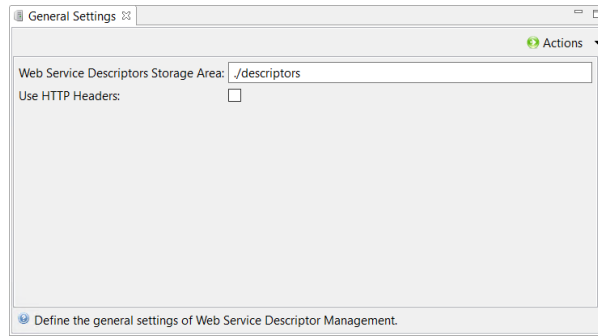


Figure 13: Configure the use of HTTP headers

**Important!** Ctrl+s will not save your changes in the server configuration editors. To save, click the **Actions** drop-down menu and select **Save**.

**Reference**  
For more information about HTTP headers in the JSON output, refer to [Testing the execution of the web service on page 108](#).

## 9.5 Configuring the Mail Sender Settings

Access ARCAD-API General Functions → ★ Mail Configuration

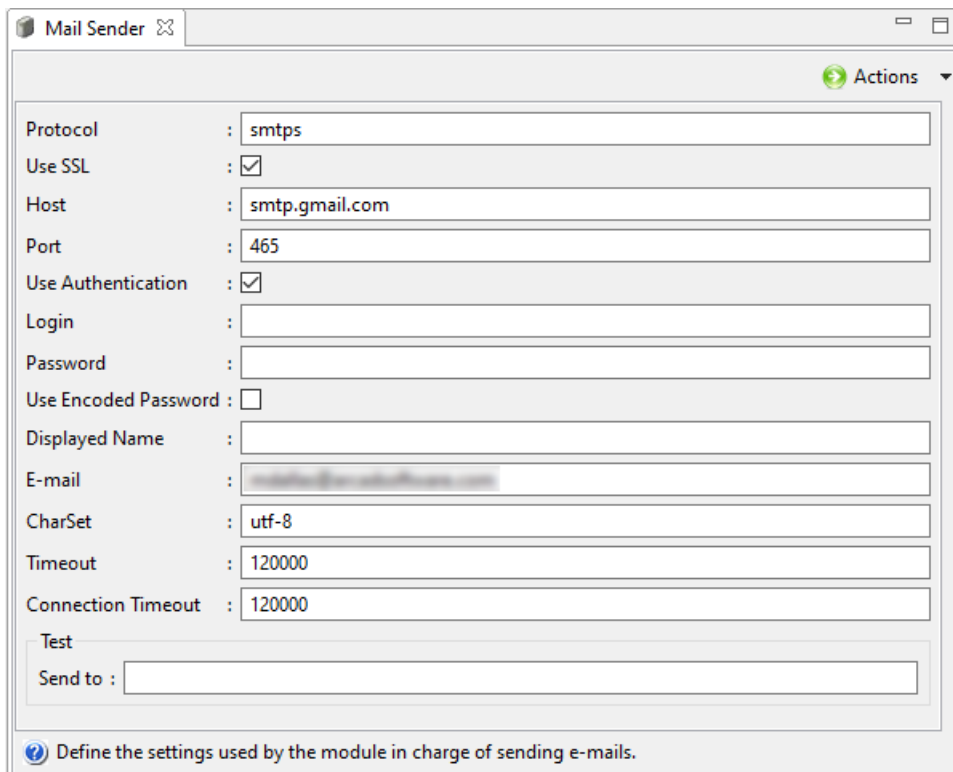


Figure 14: Configure the mail sender settings

These settings are used when an automatic email is sent, if the project allows it.

**Important!**

The use of the SMTPS protocol is highly recommended to ensure the secure transfer of notification messages.

**Transport/Protocol**

The transport protocol used to reach the server.

**Use SSL**

Tick to use the SSL connection.

**Host**

The DNS or the IP Address of the SMTP server.

**Port**

The communication port used to reach the SMTP server.

**Use Authentication**

Authenticated connection required.

**Login / Password**

The Login / Password used to connect to the SMTP server.

**Use Encoded Password**

Tick to use encoded password.

**Displayed Name**

The name that will be displayed in the *From* field of the sent email.

**Email**

The email address that will be displayed in the *From* field of the sent email. This value is skipped if the **Displayed Name** is defined.

**Charset**

The charset used to encode non US-ASCII characters contained in the email subject.

**Timeout**

The communication timeout value in milliseconds.

**Connection Timeout**


The connection timeout value in milliseconds.

To test that the mail sender has been properly configured, enter an email address in the **Test** field and restart the server. If the parameters are correct, a number of test emails will be sent to the address entered every time the server is started.

**Important!**

If you do not want to receive an email every time the server is started, remove the addresses entered in the **Test** field.

**Important!**

Ctrl+s will not save your changes in the server configuration editors. To save, click the  **Actions** drop-down menu and select **Save**.

## 9.6 Registering License Keys

Access Configuration Categories → Administration → License Key

Activation Keys for ARCAD-API are required to set up the ARCAD-API process. Enter the **full** license key, including dashes, in the field provided.

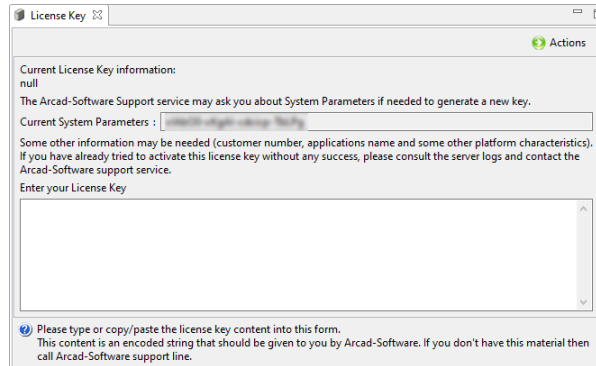



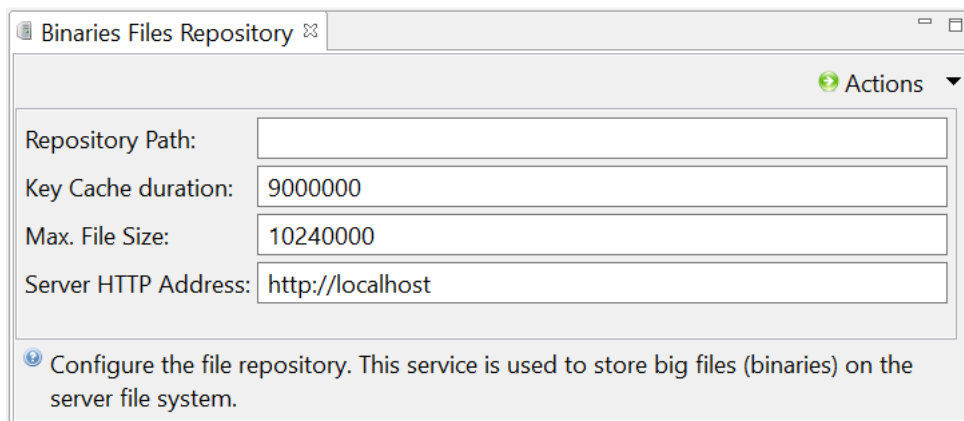
Figure 15: Register the ARCAD-API license key

### Important!

Ctrl+s will not save your changes in the server configuration editors. To save, click the  **Actions** drop-down menu and select **Save**.

## 9.7 Configuring the binary files repository

Access Administration → Binary Files Repository



This configuration section allows the storage of large binary files on the server file system.

### Repository Path

Sets the local path where the binary files are stored. This path can be relative to the application home directory.

If the field is left empty, the default repository is `./files/bin/`.

### Key Cache duration

Sets the limit, in millisecond, of the temporary public key to a file. Once this limit is reached, the file is no longer available with this key and a new one must be generated.

By default, the value is set to 9,000,000 milliseconds.

### Maximum File Size

Sets, in bytes, the maximum file size allowed to be uploaded through the HTTP server set below. By default, the value is set to 10,240,000 bytes.

### Server HTTP Address

Sets the HTTP server address hosting the file repository.

This allows the local application to delegate the management of binary files to another server, but most of the time this server is the same as the web-services server.

The default server address does not contain a port number and is automatically changed into the real web-service server name.

By default, the address is **http://localhost**.

## 9.8 Working with database scripts

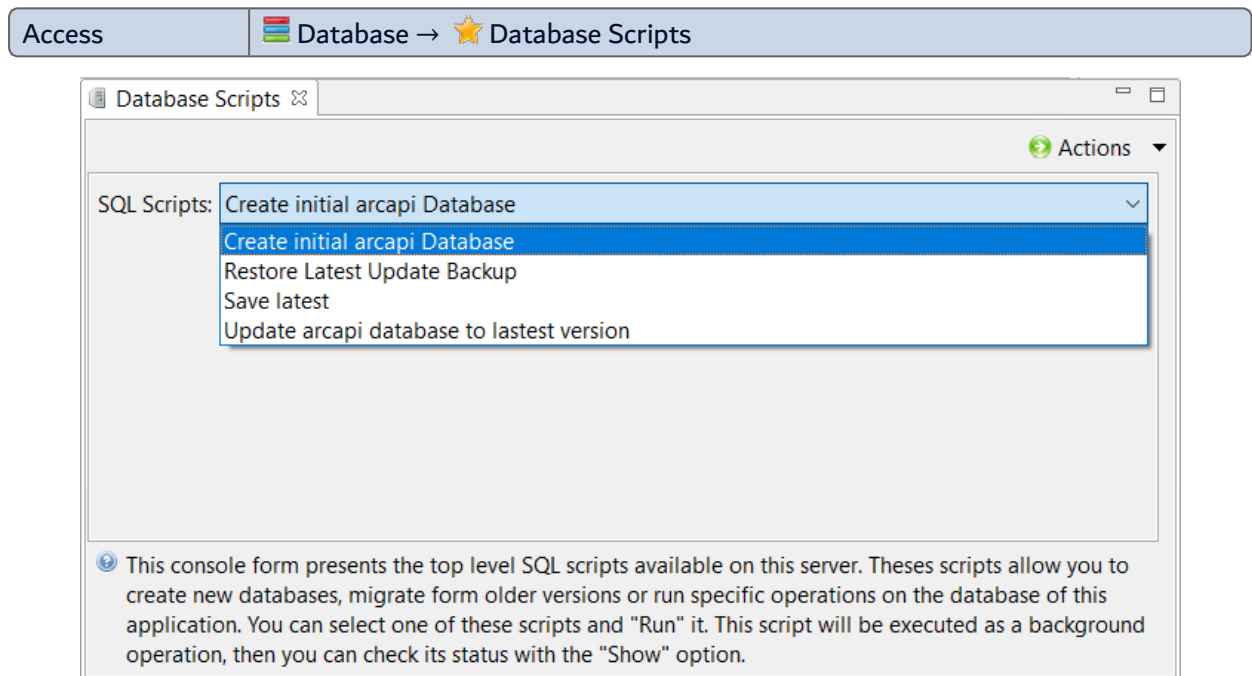


Figure 16: The H2 database maintenance scripts


The ARCAD-API Server comes with a set of scripts for its integrated H2 database. They are top-level SQL scripts that allow you to perform maintenance actions on the database.

**Step 1** Access the server's scripts.

1. Connect to the updated ARCAD-API Server.
2. Open the **Configuration** view. You may need to right-click on the server to find this option if it isn't in the navigator's toolbar.
3. Select **Database > Database Scripts** to open the **Database Scripts** editor.


**Step 2** Select the database script from the **SQL Scripts** drop-down.

**Step 3** Click the **Actions** drop-down menu and select **Run**.

**Result** Scripts are executed as a background operation. To check its execution status click the  **Actions** drop-down menu and select **Show**.

### Create initial ARCAD-API database

This script creates an empty H2 database in the server.

 **Warning!**  
This script will overwrite the H2 database already existent in the server.

### Restore latest update backup

This script restores the latest update backup that was created for the H2 database.

### Save latest



This script saves the latest version of the H2 database on the server.

### Update ARCAD-API database to latest version

This script performs an update of the H2 database after a server update.

### Update missing required data

This script allows the addition of required data or project options.

 **Important!**  
Ctrl+s will not save your changes in the server configuration editors. To save, click the  **Actions** drop-down menu and select **Save**.

## 9.9 Defining the Domain Aliases

Access  Meta-Data →  Domain Aliases

This configuration section displays the Meta-Data domain names that have aliases. These domain aliases are separated by white spaces.

Alias names can only contain alpha-numeric characters. You can edit the list, add or remove existing aliases, or add a new alias to the list.

 **Note**  
To completely remove a domain alias, remove all the values from the list.

### Domain "mem:rights" aliases

Sets the aliases for this domain.  
By default, the alias already set is **rights**.

### Domain "jdbc:arcapi" aliases

Sets the aliases for this domain.  
By default, the alias already set is **arcadpi userdb**.

# 10 Preferences

## Chapter Summary

- 10.1 Defining the default web service categories ..... 56
- 10.2 Defining mapping tables preferences ..... 57
- 10.3 Managing profiles and rights ..... 58
- 10.4 Accessing your profile ..... 59
- 10.5 Changing your password ..... 60

To manage the preferences in the ARCAD-API Studio, click the  **Preferences** icon in the navigator.

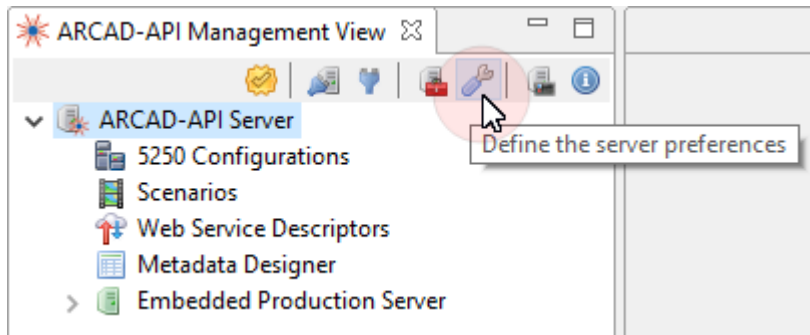


Figure 17: Manage the ARCAD-API Studio preferences

## 10.1 Defining the default web service categories

Create, edit and delete categories for the web services. Categories group the Web Service Descriptors into sets. Descriptors can be added to categories when they are created.

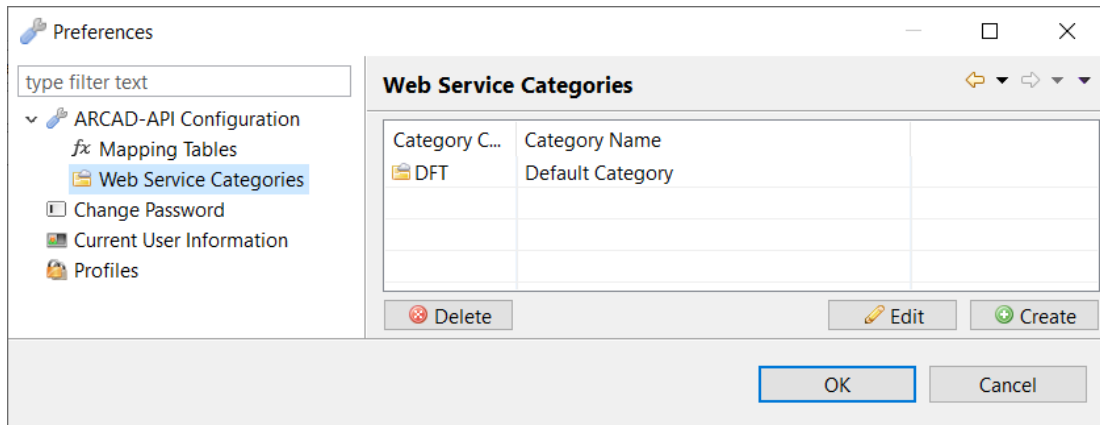




Figure 18: Define the default web service categories

Click the  **Create** button to create new categories. Each category must have a short code and a name. New categories are automatically available when creating new descriptors and also in existing descriptors' editors.

Select an existing category and click the  **Edit** button to modify its code or name. Modifications are automatically and immediately taken into account.

Select an existing category and click the  **Delete** button to remove it.



### Warning!

If any descriptors are actively saved in a category, the category is "in use" and cannot be deleted. You must first move the descriptors from the category, or delete them, then you can delete the category.

### Reference

Categories are used when creating and editing web service descriptors. Refer to [Web service descriptors on page 92](#).

## 10.2 Defining mapping tables preferences

Sometimes, the values captured on the screen don't match exactly what the web service consumer expects. Use mapping tables to map a given value to another one before returning it. Create a mapping table defining the way a field value should be returned. You assign the mapping table to a field or a column to ensure that the value returned is translated.

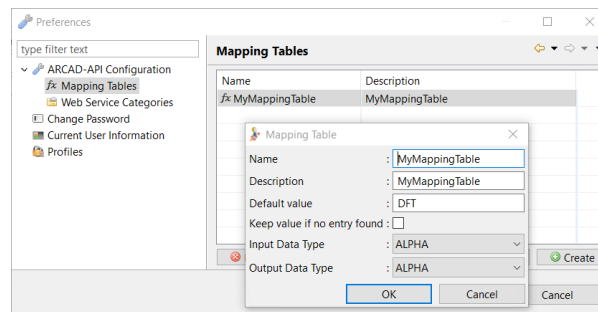


Figure 19: Define the mapping tables

To create a mapping table, click the **Create** button.

In the **Mapping Tables** dialog, define the mapping table's properties.

### Name

The name of the mapping table.

### Description

A brief description of the mapping table.

### Default value

The mapped value used if no mapping entry has been found (if the **Keep value if no entry found** option is not selected).

### Keep value if no entry found

Check this box to keep the captured value unchanged if no mapping entry has been found.


### Input Data Type

The data type of the captured value.

### Output Data Type

The data type of the mapped value.

Click **OK** to save.

To modify its parameters' values, select an existing mapping table and click the  **Edit** button. Modifications are automatically and immediately taken into account.

To manage the mapping values, click the **Values** button. In the dialog, click the  **Add** button to add the **Original Value** and its corresponding **Translated Value**. Click **OK** to save.

Select a value and click  **Edit** to modify it or  **Delete** to remove it from the mapping table.

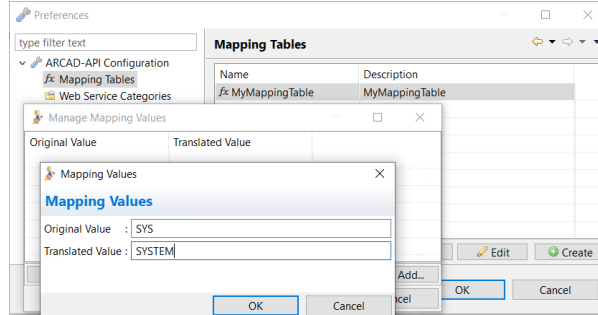


Figure 20: Define the mapping values

To delete a mapping table, select an existing mapping table and click the  **Delete** button.

### Reference

Refer to [Web service descriptors on page 92](#).

## 10.3 Managing profiles and rights

Rights grant users the possibility to view, create and modify various ARCAD-API entities. If the proper access rights are not granted to a profile, access to the corresponding entity (ex: engines) or action (ex: editing engines) is restricted.

Rights are granted to users via profiles. A profile is packed with rights that determine which items a user should be able to access throughout ARCAD-API and if they can edit them or not. Profiles are assigned to individual users.

There is no limit on how many profiles can be created. If all of your users should have access to all entities in the ARCAD-API Studio, there may be no need to create new profiles. Assigning the **All Right** profile to all users will enable everyone to have access to everything.

Open the **Profiles** menu to access and define the list of existing profiles.

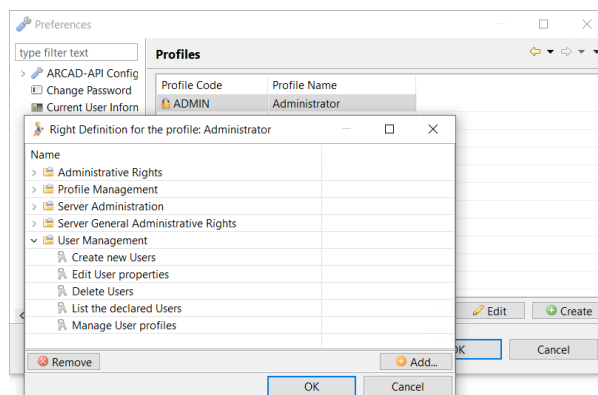


Figure 21: Defining Profiles and Rights

### 10.3.1 Defining profiles

Follow the subsequent steps to define a user profile.

To create a new profile click the  **Create** button. Fill in the required code and name in the dialog.



Defined profiles are available in the **Preferences** menu and in the **Profiles** tab of the **User** editor.

Select a profile and click the  **Edit** button to change the defined code and/or description.

To delete a profile select it and click the  **Delete** button.

Click **OK** to save.

### 10.3.2 Assigning rights to profiles

To grant rights to a profile, select it from the list and click the  **Rights** button. If the profile already has rights assigned to it, they are listed in **Right Definition** dialog. Click the  **Add...** button to display the list of **Available Rights** which contains all of the rights not yet assigned to the profile.

The rights are categorized by ARCAD-API entity or by action. Select the individual right(s) intended for the chosen profile (Ctrl+click to select multiple rights) then click **OK** to add them to the list of that profile's rights.

Remove rights granted to a profile by selecting them from the list and clicking the  **Remove** button.

Click **OK** to save.

#### **Reference**

For more information about the different rights available in the ARCAD-API Studio, refer to [User rights on page 68](#).

### 10.3.3 Assigning profiles to users

After a profile is created, it is available to be assigned to individual users. It is recommended to define the rights for each profile before assigning them to users in order to best organize the authorization hierarchy. If the profiles are well-defined in advance, it will be easier to know which user(s) to assign based on their intended tasks.

#### **Reference**


For more information about creating users or assigning profiles to users, refer to [Users on page 63](#).

## 10.4 Accessing your profile

Rights grant users the possibility to create, view and modify entities. Profiles are defined and granted rights by the administrator.

If you do not have the rights to access a certain ARCAD-API entity, when you try to, an error message displays informing you that you are not allowed to use that feature.

If you have the rights to access the **Users** view, you can see which profile(s) your user is assigned in the **User** editor.

If you cannot access the **User** editor, you can still view the list of your user's current rights in the **Preferences** menu. Click  **Current User Information** to view all of the rights granted to your current user.

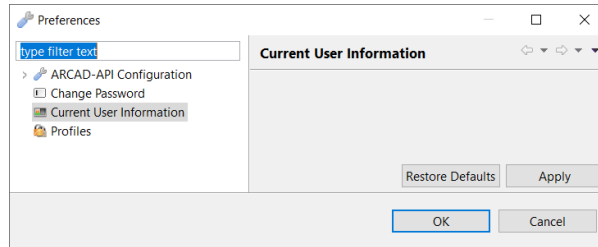



Figure 22: Current User Information (Preferences menu)

## 10.5 Changing your password

To change your user's password, right-click on the active ARCAD-API Server and select  **Preferences**. Open the  **Current User Information** node in the **Preferences** menu and select **Change Password**.

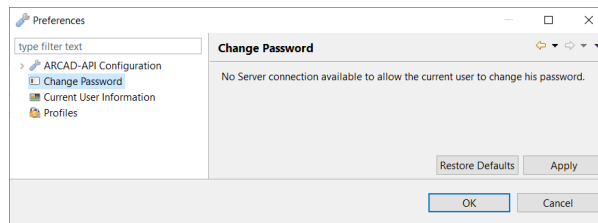


Figure 23: Change your password (Preferences menu)

Click **Apply** to push the changes live without closing the **Preferences** menu.





# USER ADMINISTRATION

# Introduction to user administration

---

ARCAD-API administrators are in charge of managing users and roles. It is important to define your team and associate the correct rights with each user, because the individuals who use ARCAD-API at different stages in the process must have accurate authority levels in order to access and, in some cases, create and/or edit the various processes.

This section covers all of an administrator's tasks associated with:

-  **User Management** (see [Users](#) on page 63)
-  **User Rights** (see [User rights](#) on page 68)

# 11 Users

Rights	User Management
Access	ARCAD-API Server → Users

## Chapter Summary

11.1 Creating users.....	63
11.2 Importing users.....	64
11.3 Editing users.....	64
11.4 Assigning profiles to users.....	66
11.5 Deleting users.....	67

User accounts define access rights for each ARCAD-API user and are managed in the ARCAD-API Studio. Each user account has a unique login.

It is important to define users and assign the correct profiles to each one. The individuals who use ARCAD-API at different stages must have accurate authority levels in order to access the various processes.

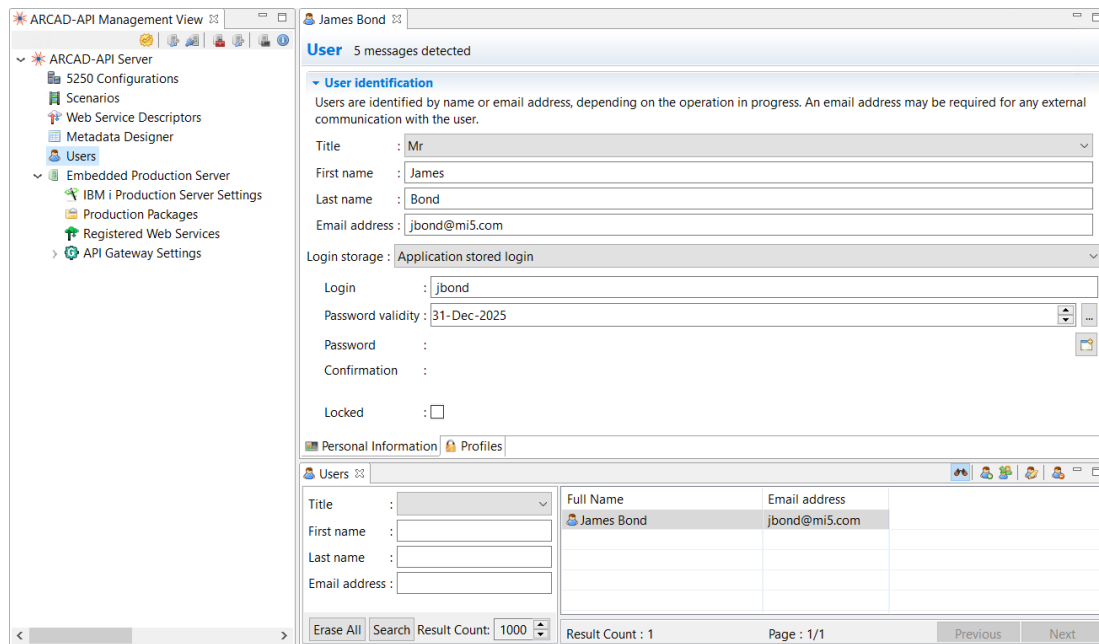


Figure 24: User Management

The **Users** view is used to manage existing users as well as to create new ones.

## 11.1 Creating users

User information can either be stored and completely managed in the ARCAD-API Server, or imported from an external LDAP which manages login credentials.

Follow the subsequent steps to create a new user.

**Step 1** To access the **Create a new user** wizard, either click the create icon in the **Users** search view or right-click anywhere in the view and select **Create a new user**.

**Step 2** Complete at least one of the fields in order to create the account.

**Step 3** Click **Finish**.

**Result** The new account is created and the **User** editor opens automatically.

 **Important!**

It is important to continue editing the new user's information right away to complete the user configuration and assign a login and password.

## 11.2 Importing users

---

From the **Import User** view, you can import your users in batch from an LDAP login source, and keep users synchronized. Furthermore, you can assign the ARCAD-API profiles to the imported users immediately from this view.

Follow the subsequent steps to import users.

**Step 1** To access the **Import Users** view, click the  Import Users icon in the  **Users** view.

**Step 2** Tick **LDAP** to select the login source from which to import the users.

 **Important!**

The login source selection boxes are not displayed if the login sources are not already configured.

**Step 3** Select the target profile in the drop-down list. The list shows all the profiles created in the ARCAD-API Server.

**Step 4** Define the LDAP **User Filter** option to retrieve the list of users that can be imported, then click on the **Search** button.

The filtered results appear as a list.



**Step 5** Check the box corresponding to the LDAP user you want to import, then click **Import**.

**Result** The user(s) is/are imported and displayed in the  **Users** view.

## 11.3 Editing users

---

### 11.3.1 Editing a users' ID

To view or edit a user's information, connection details or profile(s), either right-click on the user then select  **Edit**, select the user then click the  edit icon or double-click. The **User** editor is opened where all of its details are managed.

 **Important!**

If users were imported via LDAP (Active Directory), their details are not managed in ARCAD-API. There is nothing to configure or edit for imported users



Edit the details displayed in the **Personal Information** tab of the **User** editor by making changes in the fields necessary.

Changing the user's ID (display name and the email address used when sending automatic emails) are simple edits.

Save the changes (, Ctrl+S or **File > Save**).

### 11.3.2 Defining a user's login credentials

If users are stored in the internal database, and not an external LDAP, their ARCAD-API log-in credentials must be defined.

#### **Important!**

If users were imported via LDAP (Active Directory), their login and password are not managed in ARCAD-API.

Select **Application stored login** from the **Login storage** drop-down list, then complete the necessary fields.

#### Login

The user name to enter when accessing ARCAD-API.

#### **Note**

Spaces cannot be used in user names.

#### Password validity

Select the date the user's login password will expire. Entering an expired date forces the user to change their password the first time they log in.

#### **Note**

The server can impose a certain level of password complexity depending on its settings. By default, a password of at least 8 characters long, and containing at least one lower case, one upper case, one number and one symbol is required. These rules can be extended by the server administrator. Refer to the ARCAD-API Server Configuration Guide.

#### Password

Either enter a password for the user or click the random password button to generate a random password. A confirmation window displays the random password generated by the system.

#### **Important!**

It is important to write down what it is before clicking **Yes** because after you accept a random password there is no way to recover it.

Any user can change his password after they access their account.

**Note**

The password field is encrypted. It will always display 6 bullets, no matter what the length of the real password is.

**Confirmation**

Re-enter the password chosen. The confirmation is automatically filled in for random passwords.

**Locked**

If this checkbox is ticked, the user cannot access ARCAD-API. A user's account is locked automatically if the user tries to access ARCAD-API too many times using the incorrect password. The administrator can lock accounts manually by ticking the checkbox.

Save the changes (, Ctrl+S or **File > Save**).

### 11.3.3 Associating users with external LDAP login credentials

Follow the subsequent steps to associate a user with external LDAP login credentials.

**Step 1** In the **Login Storage** drop down list, choose **Shared LDAP directory**.

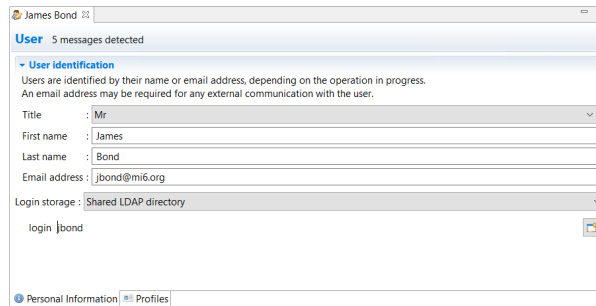


Figure 25: The LDAP user management view

**Step 2** Click the **Login** icon to select the login that will be assigned to the user. This list is prepopulated with the login candidates defined in the external LDAP directory referenced in the **Connection** configuration category.

**Result** The users can login to ARCAD-API using their LDAP login information.

## 11.4 Assigning profiles to users

At least one profile is required for every user because they grant the rights required to manipulate entities or items. If a user has no profile, they have no rights to access, create or edit anything.

Select the **Profiles** tab of the **User** editor to view and manage the user's profile(s).

To assign a profile to a user, click the  **Add** button and select the profile from the list, then click **OK**. The list of profiles is prepopulated with the profiles already available.

To remove a user's profile, select it then click  **Remove**.

**Reference**

For more information about the different rights available in the ARCAD-API Studio, refer to [User rights on page 68](#).



To view the rights granted to a profile, and for more information about creating profiles and managing the rights for each one, refer to [Managing profiles and rights on page 58](#).

## 11.5 Deleting users

---



### **Warning!**

Deleted users cannot be recovered.

To delete a user, select it in the 👤 **Users** view and click the 🗑️ Delete icon. Click **OK** to confirm or click **Cancel** to keep the user.

# 12 User rights

User roles make it possible to allow or deny access to specific features of the ARCAD-API Studio.




Role	Description
<b>Administration Rights</b>	<p>Enables users to configure the items on the API Server.</p> <ul style="list-style-type: none"> <li>• Create metadata</li> <li>• Edit metadata</li> <li>• Manage centralized internationalization</li> </ul>
<b>Profile Management</b>	<p>Enables users to access, create and manage  <b>Profiles</b> in the  <b>Preferences</b>.</p> <ul style="list-style-type: none"> <li>• Create profiles</li> <li>• Edit profiles</li> <li>• Delete profiles</li> </ul>
<b>Server Administration</b>	<p>Enables users to configure the ARCAD-API Server.</p> <ul style="list-style-type: none"> <li>• Manage server preferences</li> <li>• Manage server configuration</li> </ul>
<b>Server General Administration rights</b>	<p>Enables users to configure the ARCAD-API Server.</p> <ul style="list-style-type: none"> <li>• Access to the administration console</li> <li>• Access to the remote shell</li> <li>• Manage and edit server configuration parameters</li> <li>• Substitute users accessing web-services</li> </ul>
<b>User Management</b>	<p>Enables users to access, create and manage  <b>Users</b>.</p> <ul style="list-style-type: none"> <li>• Create users</li> <li>• Edit users</li> <li>• Delete users</li> <li>• List the declares users</li> <li>• Manage the user's profiles</li> </ul>

Table 16: The user rights



# CREATING APIS

# Introduction to creating APIs

---

APIs are created using a number of different tools. This section describes how to use each tool and what each accomplishes.

- [IBM i connections on page 71](#)
- [Metadata on page 73](#)
- [Scenarios on page 85](#)
- [Web service descriptors on page 92](#)

# 13 IBM i connections

Access  ARCAD-API Server →  5250 Configurations

## Chapter Summary

13.1 Creating 5250 session configurations.....	71
13.2 Editing 5250 session configurations.....	71
13.3 Deleting 5250 session configurations.....	72

In order to record the metadata and create the scenarios for a web service, ARCAD-API must be connected to your target IBM i server.

You can create multiple connections in the 5250 Configurations view and select the one required at the different stages of a web service creation.

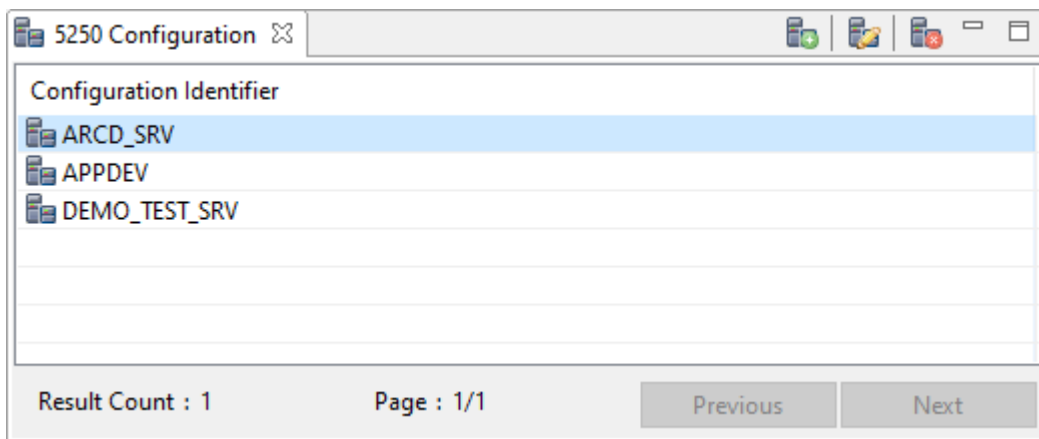




Figure 26: Accessing 5250 Session Configuration search view

5250 session configurations are accessed and managed in the  **5250 Configurations** search view. To open this view, click on the  **5250 Configurations** node in the navigator.

## 13.1 Creating 5250 session configurations

Follow the subsequent steps to create a 5250 session configuration.




**Step 1** Either click the  create icon in the toolbar or right-click and select  **Create**.

**Step 2** In the wizard, enter a name for the connection that you will recognize in the **Configuration ID** field. Then define the **Host Name** (the IP address) and partition's **CCSID** and **Session Name**.

**Step 3** Click **Finish**.

**Result** The configuration is created and ready to be used when connecting to the server to create metadata and record scenarios.

## 13.2 Editing 5250 session configurations

To edit 5250 session details, either right-click on it in the  **5250 Configurations** search view then select  **Edit**, select the item then click the  edit icon in the toolbar or double-click it.

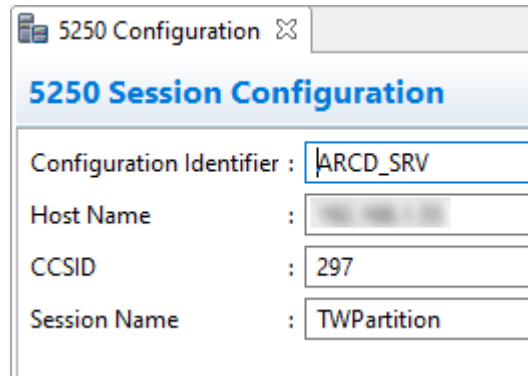


Figure 27: Editing 5250 Session Configurations

### Configuration ID

A unique string that identifies the connection. This label is used throughout the ARCAD-API Studio to select the entity.

### Host Name

Enter the IP address for the target IBM i.

### CCSID

Enter the CCSID used by the connection.

### Session Name

Enter a generic name for the connection. This information is not reproduced in the ARCAD-API Studio and is for information purposes only.




Save the changes (, Ctrl+S or **File** > **Save**).

## 13.3 Deleting 5250 session configurations

---

### **Warning!**

Deleted 5250 configurations cannot be accessed or recovered.

To delete a 5250 configuration, either right-click on it in the  **5250 Configurations** search view then select  **Delete**, or select the item then click the  delete icon in the toolbar. Click **OK** to confirm or click **Cancel** to keep the 5250 configuration.



# 14 Metadata

Access  ARCAD-API Server →  Metadata Designer

## Chapter Summary

14.1 Accessing the Metadata Designer.....	74
14.2 Creating metadata.....	75
14.3 Duplicating metadata.....	83
14.4 Deleting metadata.....	84

The 5250 flow does not contain any functional information about what is displayed on screen. It only contains information about what must be displayed and how it must be displayed (character, display attributes, etc.). APIs must be able to interpret data displayed on the 5250 screen. To do this, you must assign IDs to the areas on your screen that contain data that the web service needs to interpret.

ARCAD-API allows you to assign functional attributes to specific areas in the 5250 screen. A screen area is a rectangle drawn on screen that delimits the portion of text to identify. This means that you can select a certain area in the emulator, that corresponds to a field in your application, and attribute an ID. These IDs are called metadata. Metadata identify one or more screen area(s).

Metadata IDs are used when designing scenarios for your web service. When a web service calls a metadata ID, the data contained in the related screen area(s) can be extracted from the 5252-flow and interpreted or manipulated if there are values to input in a given field.

### Example

The Customer ID in your application is displayed on line 4, columns 5 to 10 of a given screen. The only information you can get at the 5250 flow level is the character sequence and the related display attributes. You need to create an ID that allows you to know where the Customer ID is found in your emulator on each given screen.

To assign a functional ID to an area of a screen, use the metadata designer to draw a rectangle around the area and assign a metadata ID.

### Example

Two different users can run the same scenario using two different user accounts because the user and password fields are metadata. When executing the scenario via the web service, they each enter their own user/password and, via the metadata ID's, the system automatically replaces the scenario's values with the input values.

ARCAD-API can assign different types of metadata:

- **screens:** a screen ID is created by one or more screen areas that allow you to identify a unique screen. The combination of the areas defined for a screen ID and their relative positions must be unique for each screen. This means that *this* combination of areas can only be found on *this* screen. The screen ID is important because it is the parent ID for all other metadata.
- **fields:** a field ID defines a specific screen area that contains individual data.
- **lists:** a list ID defines a portion of the screen that is used to display a list of items as a table. A list's area must cover the *entire* portion of screen that is used to display the elements of the table.
  - **columns in lists:** a column ID defines the area that covers one column in a list. The column's area must be included in the area covered by the parent list ID.

- **end of list messages:** an end of list ID defines the portion of the screen where a message appears when the end of the list is reached. The text value that will be displayed when the end of the list has been reached while browsing the entire element of the table may not be found on the same screen as the parent list if the list flows over multiple screens.

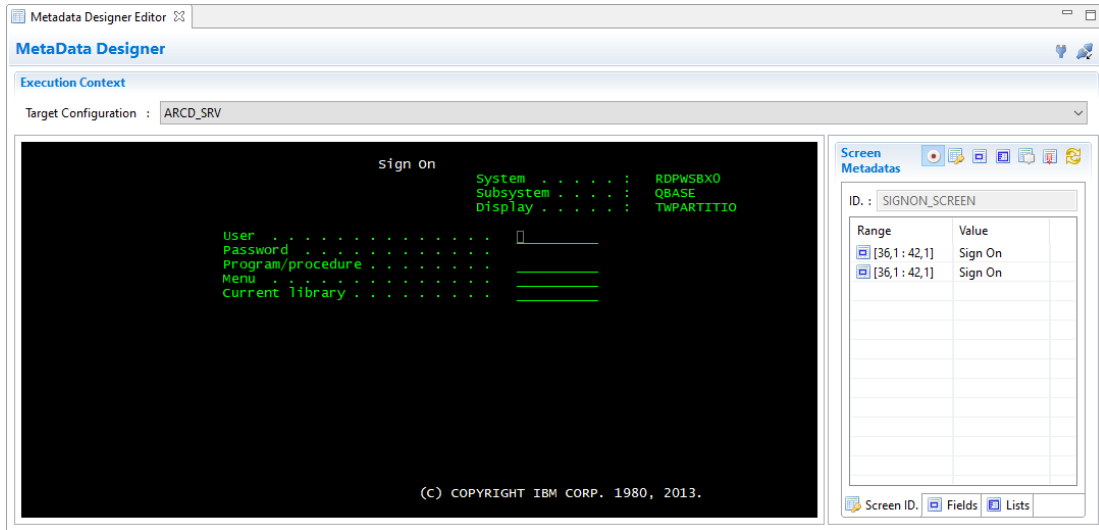


Figure 28: The Metadata designer

The metadata designer allows you to assign IDs in advance. Pre-defined metadata are available in the scenario designer. Metadata can also be created while you are recording scenarios in the scenario designer, however creating them in advance via the simplified metadata designer may save you time. There is also a possibility to duplicate metadata that are reused from on screen to another.

**Note**  
The F2, F5 and F12 keys can be used inside the emulator because they have been removed from the standard key binding context.

## 14.1 Accessing the Metadata Designer

The **Metadata Designer** is accessed directly from the navigator. There is only one screen for this designer which you use to access any of the IBM i connections available.

The designer must connect to an IBM i server, therefore at least one target 5250 session connection must already be defined.

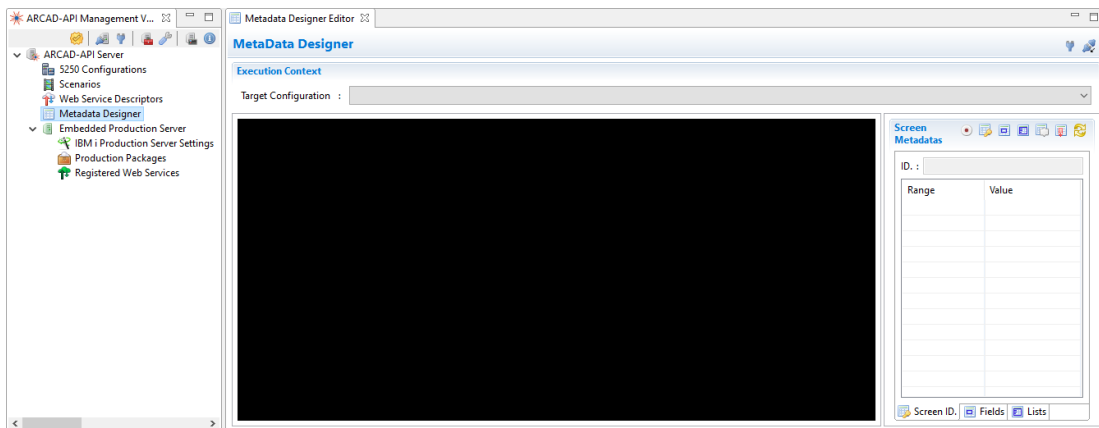




Figure 29: Accessing the Metadata Designer

To connect to the IBM i on which to capture the metadata, select the partition connection from the **Target Configuration** drop-down list. This list is pre-populated with all of the 5250 Session Configurations that are already defined. The connection is established automatically. The first screen in the 5250 will appear if the connection is successful.

 **Important!**

If a connection cannot be made, the screen will remain black. Make sure that your local machine and user have access to the target server in the **5250 Configuration** editor.


To  disconnect and  reconnect to any partition, click the corresponding icon(s).

 **Reference**


For more information about creating IBM i connections to use in the metadata designer, refer to [IBM i connections on page 71](#).

## 14.2 Creating metadata

Metadata is captured by drawing a parameter around specific screen areas in the 5250 interface using your mouse. Drag your cursor to define a parameter around a field in which you either enter text or in which text is automatically displayed, or around whole lists to capture all of their contents.


The metadata designer displays a 5250 emulator on which you can navigate through your application. Navigate "as usual", using your keyboard, through your application to reach the screens that contain the data to identify. To navigate in your application, you must be in  Record Mode.

Once you have reached a screen to identify, switch to the mode that corresponds to the type of metadata you wish you identify. After you have identified everything you need on one screen, toggle back to record mode to continue the navigation. The process continues as such, going back and forth between entering commands to navigate the 5250 interface and then creating metadata IDs.

You must always remember to click back on the  **Toggle to Record Mode** when you are finished creating the metadata and are ready to navigate in the interface again. If you are not in record mode, you will not be able to enter commands and navigate because the system believes you are still drawing perimeters to define metadata IDs.

 **Note**

When you are defining metadata in the designer, you are not recording a scenario. Your movements and the commands you enter are not recorded. In this view the "record mode" is for working in the 5250 interface and the metadata modes are for drawing the parameters to capture information.

Once they are created, if the metadata do not appear automatically, click on the  Refresh Metadata icon to update the designer.

Follow the subsequent steps to create the metadata that you will use, and reuse often, throughout the scenarios in your web service.

### Step 1 Select the **Target Configuration**.

At first, the designer is automatically in  Record Mode and the system is ready for you to navigate through the 5250 emulator.

**Note**

In record mode, the screen ID displayed in the metadata area always corresponds to the last screen that contained defined data, not necessarily the current screen. If there is no data defined for the current screen, especially no screen ID, the ID field will not reflect the current state of what is showing on-screen.

**Step 2** Use your keyboard to log in and enter commands as necessary to navigate through your application and access the various screens that contain the metadata you require.

**Step 3** When you have reached a screen that contains data to identify, click the icon that corresponds to the type of metadata to define the various metadata. The specific information about defining the different types of metadata are described below:

- Define screen IDs below
- Define field IDs on page 78
- Define list IDs on page 79
- Define column IDs on page 80
- Define end of list IDs on page 81

**Note**

When creating a new ID, ARCAD-API verifies that there are no existing IDs that already cover the area selected. You cannot create two overlapping metadata. The data on screen can always only belong to one defined metadata ID.

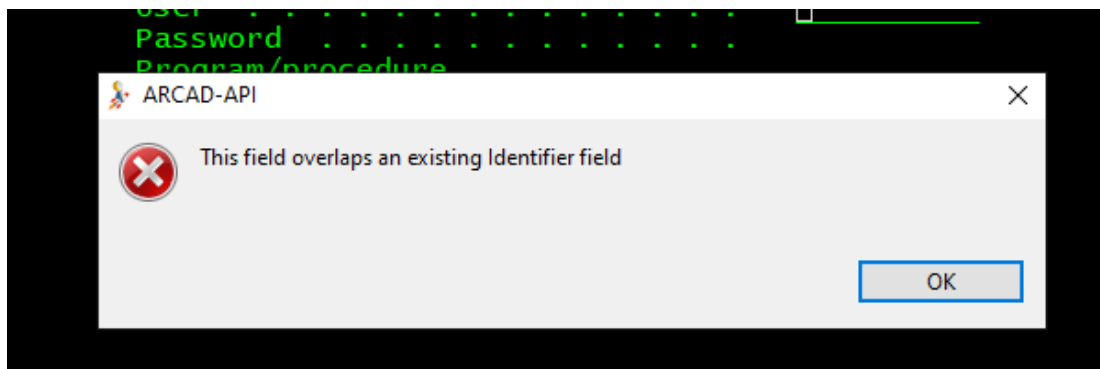



Figure 30: The overlapping field warning dialog

**Step 4** Toggle back to the  Record Mode to navigate your application and capture the metadata on different screens.

**Step 5** Repeat steps 2 through 4 as many times as you need to define all of the metadata in your application.

**Step 6** Click  Disconnect to sign off from the Target Configuration.

### 14.2.1 Define screen IDs


 Screen metadata is used to define individual, unique screens. Screen IDs are important to identify because they are the parent element for any other metadata defined on the screen. The screen ID is

associated with all of the fields and/or lists so that each metadata is truly unique. The data must conform to the information defined in the metadata *and* to the parent screen ID to be valid.

A screen can be defined by one or multiple fields. If a screen can be considered unique using only one field of information, such as the screen's title, then it is recommended that only the one area be used. However, if you have multiple screens in your application with the same title, each screen can use the title as an ID key as well as any number of other identifying fields on the screen, such as a list name. If there are multiple areas that make up a screen ID, they must all be satisfied before searching for the child metadata because all of the areas together make up the "uniqueness" factor for the screen ID.

Follow the subsequent steps to create screen ID metadata.

**Step 1** Navigate to the desired screen using the  Record Mode.

**Step 2** Click the  Screen ID icon to switch to the mode that allows you to draw the perimeter around the areas that define the screen.

**Step 3** Select the area of the screen you want to use as an identifier. Use your mouse to draw a box around the area.

 **Note**

To use multiple areas on the screen to define the screen, press and hold the SHIFT key while you draw multiple boxes.

**Step 4** Enter the ID name for the screen in the dialog that appears.

**Step 5** Click **OK** to save.

**Result** The screen ID is created and appears in the **Screen Metadata** section. The range of the screen area selected is displayed as well as the name of the metadata ID. The perimeter around the screen area selected for the metadata ID is highlighted.

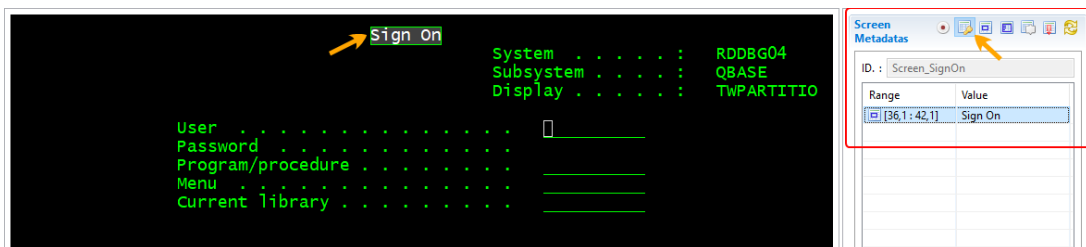



Figure 31: The captured Screen ID metadata


**Step 6** Toggle back to the  Record Mode to continue navigating.

 **Warning!**

If you remain in the  Screen ID mode and continue to click in the emulator, the system considers that you are redefining the screen ID's perimeter. The new range for the ID will replace the original range. To avoid this, it is best to habitually toggle back to record mode as soon as you have created an ID.

If you do replace the original range for the ID, you can simply redraw the perimeter as required.

## 14.2.2 Define field IDs


 Field metadata IDs define the scope of one field. You can create any number of field metadata on each screen. The only constraint is that the perimeter for each field is unique.

### Important!

You cannot create two IDs for the same area on a screen and you cannot create two IDs with the same name on the same screen. Two field IDs can have the same name as long as they are found on different screens.

Follow the subsequent steps to create field ID metadata.

**Step 1** Navigate to the desired screen using the  Record Mode.

**Step 2** Click the  Field ID icon to switch to the mode that allows you to draw the perimeter around the area that defines the field.

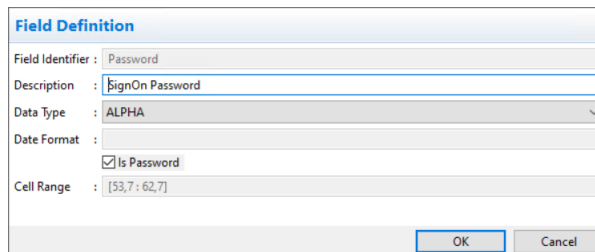
**Step 3** Select the area of the screen you want to use as an identifier. Use your mouse to draw a box around the area.

**Step 4** Enter the ID name and a description for the field in the dialog that appears.

**Step 5** Select the **Data Type** in the drop-down list. The data type can be Alpha, Integer, Boolean, Float, Date. For date data types, enter a **Date Format** in the field.

### Important!

If the field is a password, be sure to tick the **Is password** checkbox in the **Field Definition** dialog. While executing a web service that involves a password field, its content will not be displayed in the log file. Anonymous stars will be displayed, instead, protecting your identifying data.



The image shows a 'Field Definition' dialog box with the following fields and values:

Field Identifier :	Password
Description :	SignOn Password
Data Type :	ALPHA
Date Format :	
<input checked="" type="checkbox"/> Is Password	
Cell Range :	[53,7 : 62,7]

Buttons: OK, Cancel

Figure 32: Tick the Is Password checkbox for identifying data fields

**Step 6** Click **OK** to save.

**Result** The field ID is created and appears in the **Screen Metadata** section. The range of the area selected is displayed as well as the name and type of the metadata ID. When you select an ID, the perimeter around the screen area selected is highlighted and the name of the ID is displayed.

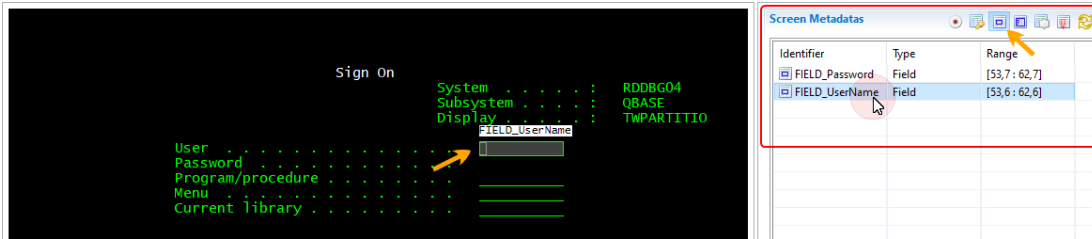


Figure 33: The captured Field ID metadata

**Step 7** Toggle back to the Record Mode to continue navigating.

### 14.2.3 Define list IDs

List metadata is used to define the entire area in which a list is and could be displayed. List IDs are important to identify because they are the parent element for any column or end of list metadata.

Follow the subsequent steps to create list ID metadata.

**Step 1** Navigate to the desired screen using the Record Mode.

**Step 2** Click the List ID icon to switch to the mode that allows you to draw the perimeter around the entire area of the list.

**Step 3** Select the area of the screen you want to use as an identifier. Use your mouse to draw a box around the area.

**Important!**

For lists, the screen area must cover the whole list, even if there are not currently enough entries to go all the way to the bottom of the page. Ensure that all the lines and columns that could potentially display an item in the list are included in the selected area.

**Step 4** Enter the ID name for the list in the dialog that appears.

**Step 5** Click **OK** to save.

**Result** The list ID is created and appears in the **Screen Metadata** section. The range of the area selected is displayed.

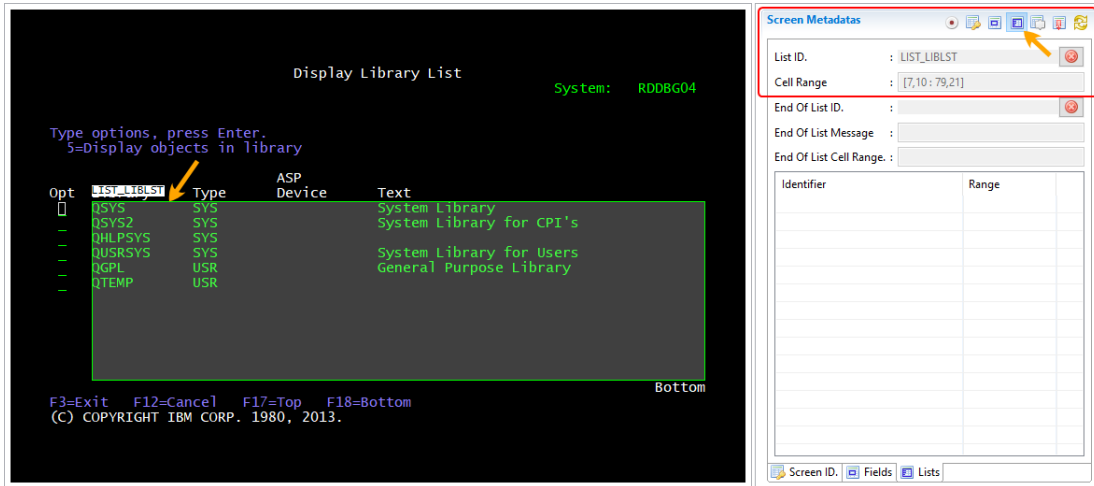


Figure 34: The captured List ID metadata

**Step 6** Toggle back to the Record Mode to continue navigating.

### 14.2.4 Define column IDs

Column metadata is used to define specific columns in lists. Each column ID must be part of a parent list.

Follow the subsequent steps to create column ID metadata.

**Step 1** Navigate to the desired screen using the Record Mode.

**Step 2** Click the Column ID icon to switch to the mode that allows you to draw the perimeter around a column in a list.

**Step 3** Select the area of the screen you want to use as an identifier. Use your mouse to draw a box around the area.

**Important!**  
For columns, the screen area must cover the whole column, even if there are not currently enough entries to go all the way to the bottom of the page. Ensure that all the lines that could potentially display a list item are included in the selected area.

**Important!**  
The perimeter drawn for a column must be inside the perimeter of the parent list. The perimeter of the list is displayed on screen to guide you.  
  
If the column is outside of the parent list, an error is displayed and you must draw the perimeter again.  
  
If there is no parent list ID, create it first.

**Step 4** Enter the ID name and a description for the column in the dialog that appears.

**Step 5** Click **OK** to save.



**Result** The column ID is created and appears in the **Screen Metadata** section as part of the parent list's details.

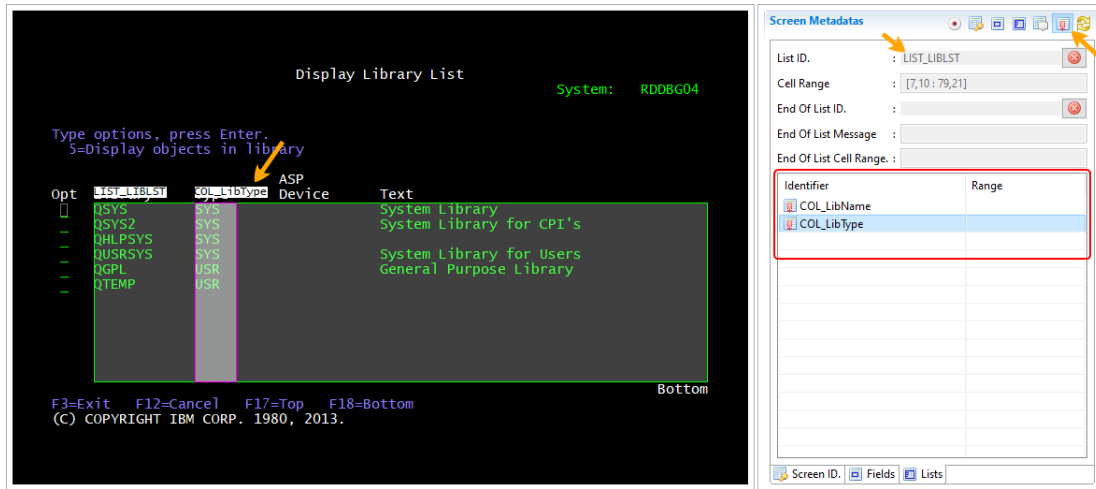


Figure 35: The captured Column ID metadata

**Step 6** Toggle back to the Record Mode to continue navigating.

### 14.2.5 Define end of list IDs

The end-of-list ID is attributed to the area where a predefined message will appear once the end of the list is reached.

Follow the subsequent steps to create end-of-list ID metadata.

**Step 1** Navigate to the desired screen using the Record Mode.

**Step 2** Click the End of List ID icon to switch to the mode that allows you to draw the perimeter around the message at the end of a list.

**Step 3** Select the area of the screen you want to use as an identifier. Use your mouse to draw a box around the area.

**Important!**  
The end of list ID must correspond to a parent list. If there is no parent list ID, create it first.

**Step 4** Enter the ID name for the message in the dialog that appears.

**Step 5** Click **OK** to save.

**Result** The end-of-list ID is created and appears in the **Screen Metadata** section. The range of the area selected is displayed.

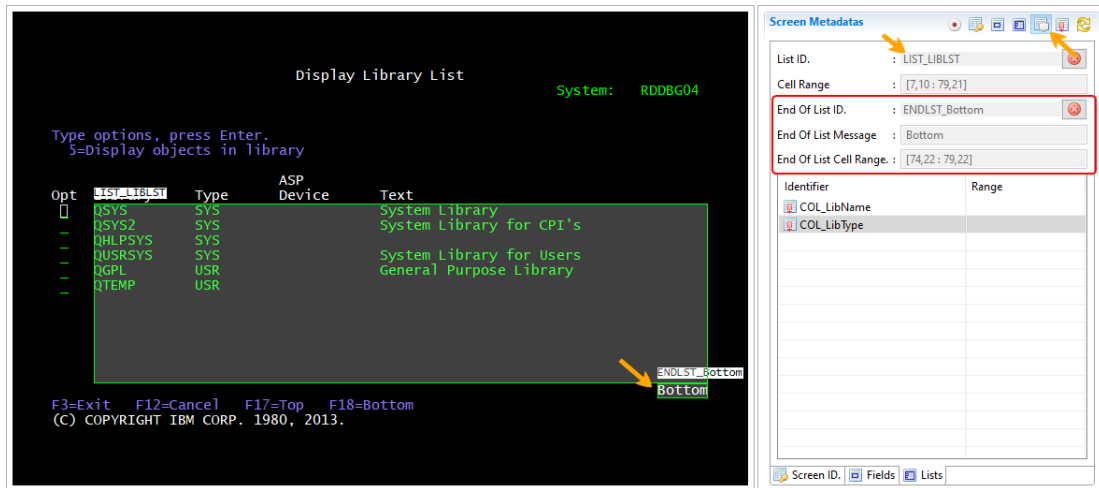


Figure 36: The captured End of List ID metadata

**Step 6** Toggle back to the  Record Mode to continue navigating.

**Note**





It is possible to reach the end of a list without defining a End of Line message.

If no End Of Line Message has been defined, ARCAD API compares the existing list content with the previous one and considers that the end of list has been reached if the current list content is the same as the previous one.

## 14.2.6 Navigating through multi-page data

To take into account columns containing hidden data in other parts of the emulator, you can scroll horizontally through columns. Select the wanted navigation mode when defining the columns in the **Metadata Designer** view, and then navigate horizontally on the IBM i emulator.

There are two navigation modes available:

- **Axial:** this navigation mode contains two navigation keys, the  Previous Page button and the  Next Page button. With these two keys, you can move left and right on the emulator.
- **Circular:** this navigation mode contains only one navigation key, the  Next page button. Once the emulator reaches the last page, the next time you press the  Next Page button, it takes you back to the initial page.

**Important!**

Make sure that you have defined your list ID beforehand.

**Reference**

For more information about the list ID definition, refer to [Define list IDs on page 79](#).

Follow the subsequent steps to set the horizontal navigation.

**Step 1** In the **Screen Metadata** section of the **MetaData Designer** view, click on the  Toggle to Screen Columns definition mode icon.

**Step 2** In the **Navigation Type** option, select one of the two navigation modes available in the drop-down list.

**Step 3** In the **Previous Page** and **Next Page** options, select the emulator keys available in the drop-down list.

After defining the navigation option and navigation keys, you can create new columns. Select the area of the screen you want to use as an identifier. Use your mouse to draw a box around the area.

**Result** The horizontal navigation is set and can be used to display hidden data columns.

### Setting metadata columns

For already existing columns, it is possible to edit their page number, as well as the navigation mode and navigation keys.

#### **Warning!**

If you enter a page number that does not correspond to a column, the web services using this column, either as input or as output, return an execution error.

Follow the subsequent steps to edit existing columns.

**Step 1** In the **Screen Metadata** section of the **MetaData Designer** view, click on the  Toggle to Screen Columns definition mode icon.

**Step 2** Double-click on an existing column to open the **Column Definition** dialog.

**Step 3** Tick the **Set Metadata Column** option to be able to set the **Column Page** option.

A confirmation dialog opens.

Click **OK** to confirm or click **Cancel** to close the dialog.


**Step 4** Set the **Column Page** option. The value is an integer.

**Step 5** Select the **Navigation Type** option, as well as the emulator navigation keys in the **Previous Button** and **Next Button** options.

Click **OK**.

**Result** The construction and execution of a web service can now take into account hidden columns.

## 14.3 Duplicating metadata

You can  duplicate the metadata that are reused from one screen to another. Transferring metadata from another screen allows to save time on similar screens/scenarios.

**Step 1** Navigate to the desired screen using the  Record Mode. Click the  Duplicate icon.

**Step 2** Select the metadata you want to duplicated in the dialog that appears.

**Step 3** Enter a new Screen Identifier for the metadata. The same name is attributed by default.

**Step 4** Click **OK**.

## 14.4 Deleting metadata

To delete an entry, find it in its parent tab (**Screen ID**, **Fields**, or **Lists**), right-click on it and then select **Delete**. To find the correct data in the tab, you must first navigate to the parent screen using the **Record Mode**.

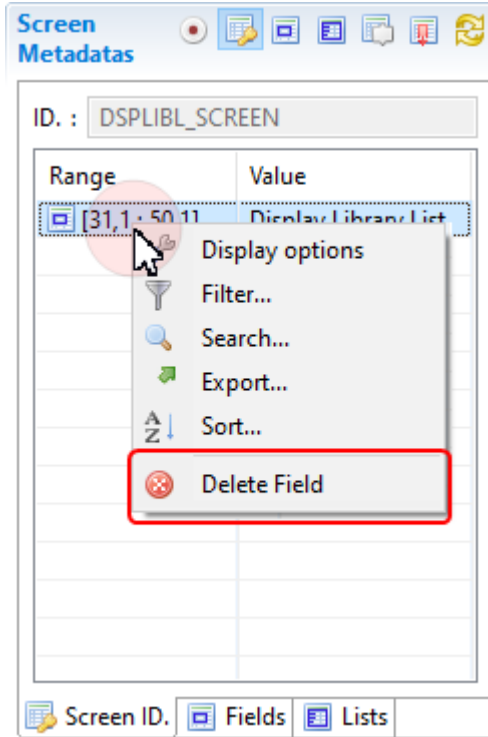


Figure 37: Deleting metadata

# 15 Scenarios

Access  ARCAD-API Server →  Scenarios

## Chapter Summary

15.1 Creating scenarios.....	86
15.2 Editing a scenario's ID.....	87
15.3 Recording scenarios.....	87
15.4 Viewing the steps in scenarios.....	88
15.5 Managing unexpected screen actions.....	89
15.6 Creating new scenarios based on existing steps.....	90
15.7 Testing scenarios.....	91
15.8 Deleting scenarios.....	91

The main goal of ARCAD-API is to be able to provide REST web services about information systems that don't provide access to the IBM i's internal data storage or processes. Often for this kind of system, the only way to access the data is via a 5250 emulator that displays a green screen interface.

ARCAD-API simulates what a 5250 user does and automatically captures the displayed data. To create this simulation, you need to first capture the process by recording it live in a 5250 emulator. Capturing a process means executing a functional process using the 5250 GUI that includes input and output actions. The set of these different steps of the execution of this process is called a scenario.

Once this scenario has been recorded, it can be replayed. When the scenario is replayed, the data displayed is captured.

Consuming an ARCAD-API web service means that the ARCAD-API Server will automatically replay the recorded scenario on the defined machine - simulating what a real user is supposed to do - and will also automatically extract the output information to return it to the web service consumer. This way, 5250 emulator-based actions can be performed via an external API.

Each time the scenario is replayed, the metadata can be modified which allows the web service to perform different actions using the same scenario by changing the input information at each execution.

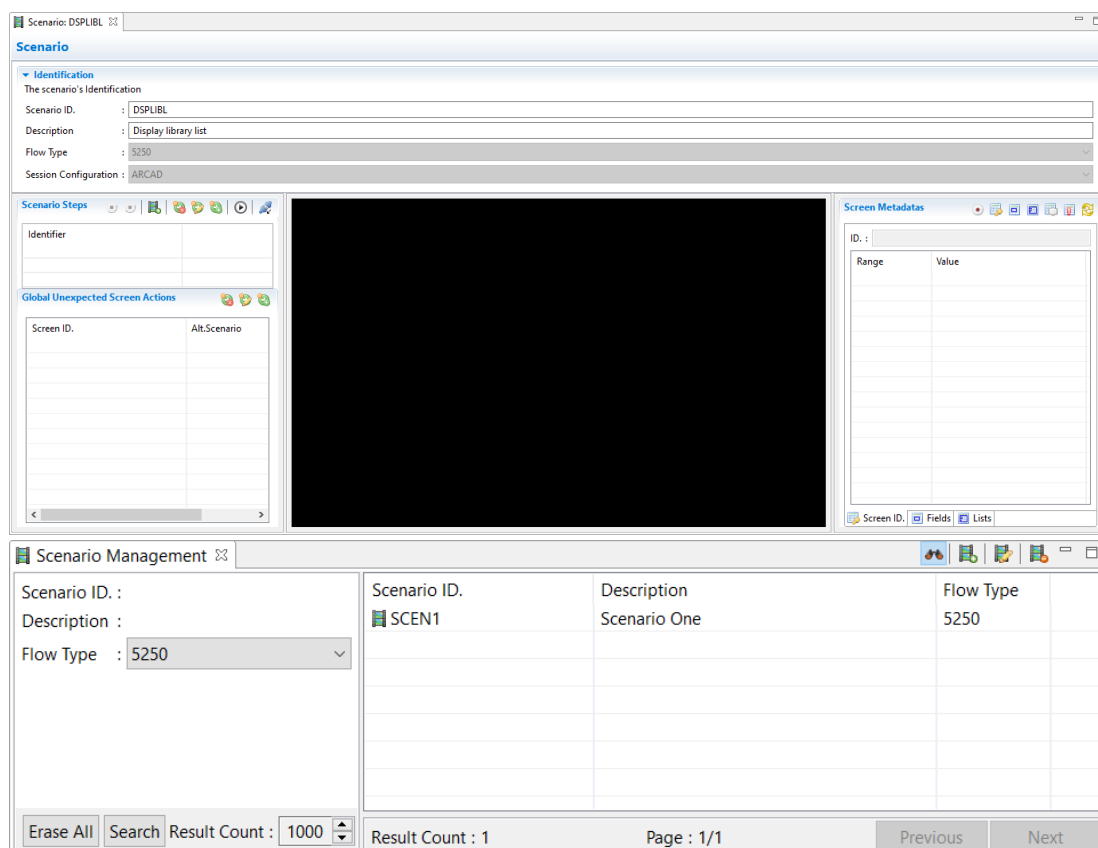






Figure 38: The Scenario editor

Scenarios are accessed and managed in the  **Scenarios** view.




## 15.1 Creating scenarios

Follow the subsequent steps to create a new scenario.

- Step 1** To access the **New Scenario** wizard, either click the  create icon in the toolbar of the  **Scenarios** search view, or right-click anywhere in the list and select  **Create**.
  - Step 2** Define the **Scenario ID** and **Description**. These values are required to create a new scenario but can be edited later.
  - Step 3** Select the **Flow Type**. This detail is not editable once the scenario is created. Today, only one flow type exists to be used when recording actions on an IBM i 5250 emulator.
  - Step 4** Select the **Session Configuration** to connect to. This is the IBM i server on which you will record the scenario. This detail is not editable once the scenario is created.
  - Step 5** Click **Finish**.
- Result** The new scenario is created and is displayed in the list in the  **Scenarios** search view. You can now use the scenario to record the process to run via your future web service.

## 15.2 Editing a scenario's ID

---

To edit a scenario's details, either right-click on it in the  **Scenarios** search view then select  **Edit**, select the item then click the  edit icon in the toolbar or double-click it.

Each scenario's editor includes the tools to edit its details and record and manage the steps carried out during the process.

### Scenario ID

A unique string that identifies the scenario. This label is used throughout the ARCAD-API Studio to select the entity.

### Description

A detailed description of the scenario to help you recognize it easily.

### Flow Type

The flow type defines which kind of terminal is to be used as the target configuration. The only option available today is IBM i 5250.

### Session Configuration

Select one of the 5250 session configurations in order to access the IBM i partition in the scenario designer. The list is prepopulated with the list of sessions available in the **5250 Configurations** view.



#### Reference

For more information about creating connections to 5250 partitions, refer to [IBM i connections on page 71](#).

Save the changes (, Ctrl+S or **File > Save**).

## 15.3 Recording scenarios

---

Once you have created a scenario you can begin to record the events to include in the web service.

Scenarios are sets of multiple steps carried out in a specific order to achieve a given goal, such as retrieve certain data, or execute a specific process.

Each step in a scenario is associated with a screen in the 5250 interface, the data input into that screen and a validation key that is used to transmit the content of the screen.

Open the scenario editor to record and manage the steps in a scenario.

**Step 1** Click the  **Start recording the scenario** button to begin.

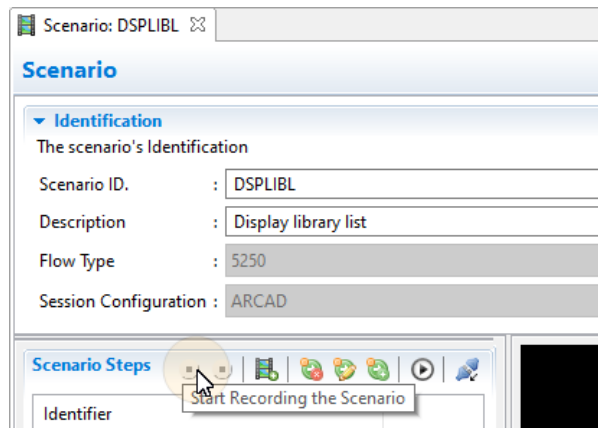


Figure 39: Start recording the scenario

Once you start recording, the system records all of the actions you perform in the interface.

- Step 2** Log on to your session and navigate through your application accomplishing the task(s) as defined for your scenario by entering commands as required.

Each time you change screens a new step is automatically created in the list of steps.



**Tip**

Every action entered while you record is taken into account. It is recommended to "practice" your scenario in advance so you know where to go from start to finish. This will reduce the noise in your scenario and limit the number of superfluous screens and actions recorded. The cleaner your scenario is while recording, the fewer unexpected data you will have to clean up after recording is finished.

- Step 3** [Optional] Metadata for each screen can be created when recording scenarios. However, it is recommended that you define all of the metadata you may need for each of your screens in advance, using the metadata designer. This way, you don't need to stop while you are recording your scenario to define the screens and fields as you go. If you don't have to define the screen's information, you can record the scenario fluidly and therefore avoid injecting too many unnecessary unexpected actions in the scenario.

If you choose to define screens and/or fields while you record scenarios, you do so the exact same way that is described in the [Creating metadata](#) chapter.



**Reference**

For more information about creating metadata in advance, refer to [Metadata on page 73](#).

- Step 4** Click  Disconnect to sign off of 5250 session and finalize the scenario.

- Step 5** End the recording by clicking the **Stop recording the scenario** button.

Save the changes (, Ctrl+S or **File > Save**).

**Result** The scenario has been recorded. You can now test it or use it to create a descriptor.

## 15.4 Viewing the steps in scenarios

After your scenario has been recorded you can review the different steps that were recorded.



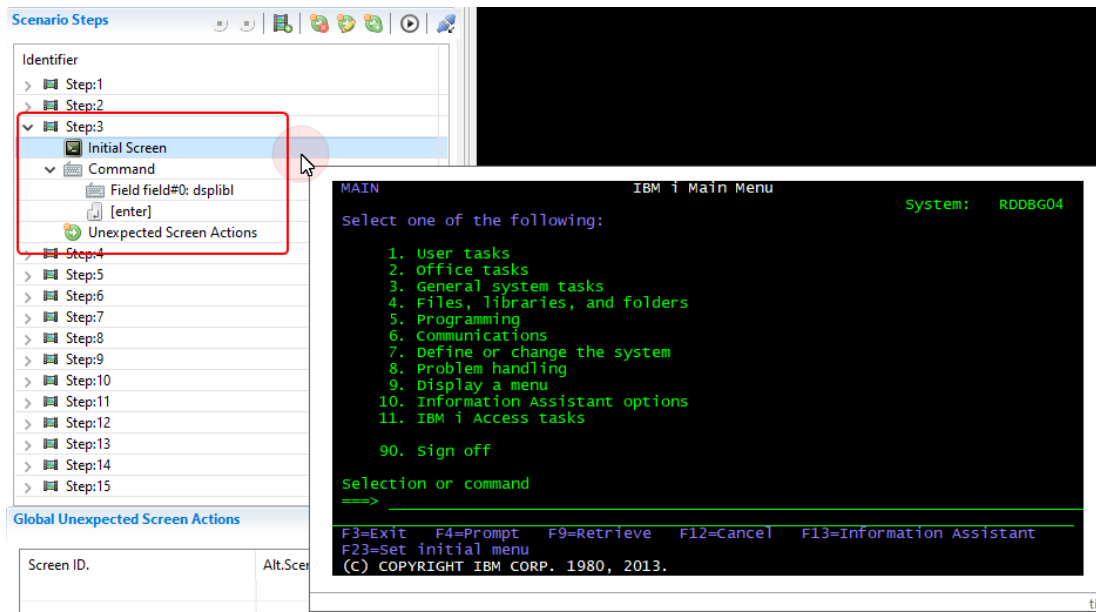


Figure 40: Viewing the steps in a scenario

Each step in a scenario has three items:

- The initial screen on which the step began. Click on the icon to display an image of the screen.
- The command or data entered on the screen, which includes the fields in which the command(s) was/were entered, the text entered and finally the command or action used to validate the input (such as the "press enter" action).
- Any unexpected screen actions

## 15.5 Managing unexpected screen actions

You may find that actions were recorded in your steps that don't produce the output that your scenario expects. You can manage how the scenario should react when it is replayed if it encounters a screen other than the one it expects (i.e. the screen in the following step or the screen that the actions should normally produce). An unexpected screen action is an action that will be executed if the screen sent by the server is not the one the scenario expected.

If an unexpected screen appears at a given step, the ARCAD-API engine will retrieve any instructions that have been inserted into the scenario at this level and will try to execute them. The system can be instructed to run a secondary scenario when it encounters a given screen or to skip the step the produced the unexpected screen.

### Important!

The most important thing to ensure when diverging from the established scenario is to create a path from the alternate route back to the current execution. An alternative scenario is a scenario that defines the steps to be executed starting from the unexpected screen and ending at the exact next step of the current scenario. In other words, the alternative scenario must lead back to the exact next step so that the process can branch back into the current execution.

An unexpected screen action is defined by a screen ID and an action to accomplish when the screen is displayed. This means that each unexpected screen must have a screen ID metadata associated with it.

 **Reference**  
Metadata on page 73.

Follow the subsequent steps to add unexpected screen actions to a scenario.

**Step 1** Locate the step in the scenario that produced, or might produce, an unexpected screen.

**Step 2** Select the  **Unexpected Screen Actions** option for that step.

**Step 3** Select the Screen ID associated with the unexpected screen from the **IF this screen occurred** drop-down list.

**Note**  
If the screen doesn't have a screen ID metadata associated with it yet, save your scenario, create the metadata and return to continue editing the step.



**Step 4** Define what the scenario should do when the unexpected screen is displayed:

Either select the alternative scenario to replay from the **THEN execute this scenario** drop-down list, or check the **Skip this step** box to simply not execute the command that produced the unexpected screen.

**Note**  
If you have not yet recorded the alternate scenario to replay, save your scenario, create the new scenario and return to continue editing the step.

**Step 5** Click **OK** to save.

**Result** The action to take if the step encounters an unexpected screen is added to the step.

Select the action and click the  edit icon to modify it. To delete the action, select it and click the  delete icon.

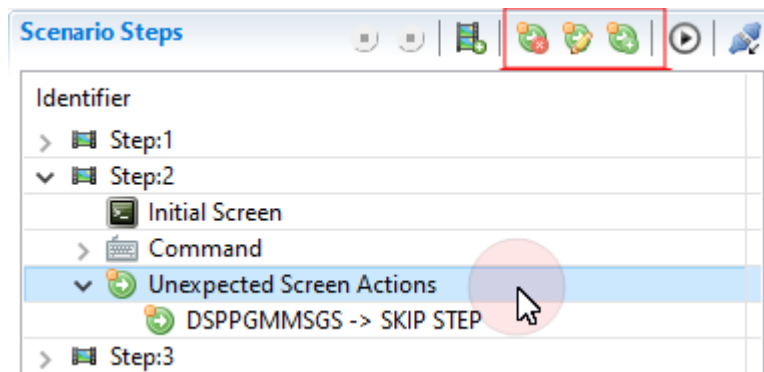


Figure 41: Adding unexpected screen actions to a step

## 15.6 Creating new scenarios based on existing steps

This process is a useful substitute for deleting steps from a recorded scenario. Recorded steps cannot be deleted but you can select and extract the steps you wish to keep. The new scenario will have the selected steps only.

You can also extract steps recorded during the creation of a scenario that could be considered as unexpected screens and actions. Then, the new scenario can be used in similar contexts as an unexpected action's scenario.

 **Note**

You cannot add more steps to a scenario created from existing steps but you can still manage unexpected actions.

**Step 1** Locate the step(s) in the scenario that has to be retained.

**Step 2** Select the  **Create new scenarios based on existing steps** option for that step(s).

**Step 3** Define the **Scenario ID** and **Description**. These values are required to create a new scenario but can be edited later.

**Step 4** Click **Finish**.

**Result** The new scenario is created with extracted step(s).

## 15.7 Testing scenarios

---

You can test your scenario directly in the recorder view at any time by clicking the **Replay the scenario** button. This is especially helpful while you are editing it because you can see if the scenario runs correctly.




This will replay the scenario in the same editor so that you can ensure your process was captured and all of the data you require is included.

## 15.8 Deleting scenarios

---

 **Warning!**

Deleted scenarios cannot be accessed or recovered.

To delete a scenario, either right-click on the item in the  **Scenarios** search view and select  **Delete**, or select the item and click the  **Delete** icon in the toolbar. Click **OK** to confirm or click **Cancel** to keep the scenario.

# 16 Web service descriptors

Access  ARCAD-API Server →  Web Service Descriptors

## Chapter Summary

16.1 Creating web service descriptors .....	93
16.2 Defining 5250 web service descriptors .....	94
16.3 Defining SQL web service descriptors .....	103
16.4 Defining REST web service descriptors .....	106
16.5 Testing the execution of the web service .....	108
16.6 Exporting web service descriptors .....	110
16.7 Exporting web service descriptors' Open API specifications .....	112
16.8 Promoting descriptors to the production server .....	112
16.9 Promoting entire descriptor categories to the production server .....	113
16.10 Deleting web service descriptors .....	114

There are three types of web services: 5250 web services, SQL web services, and REST web services. Each web service has its own specificity that are defined with the web service descriptor.

A **5250 web service descriptor** is where you define which fields/columns in your scenario are input or output data. Select the fields and columns used throughout the web service's scenario and either assign them as output data to recover the values returned when the scenario is replayed, or define them as input data so that values can be inserted into the fields to modify the scenario.

An **SQL web service descriptor** is where you define an SQL query that extracts data from the IBM i database and returns the result as JSON content.

A **REST web service descriptor** translates 5250 web services into RESTful web services, with REST API URLs exposing resources and supporting the standard HTTP methods operations (CRUD). This type of web service descriptor does not directly manage and provide the operations but it maps the CRUD operations to existing unitary web service descriptors. The execution of each operation is delegated to a unitary 5250 web service descriptor even if the consumer is not aware of it.

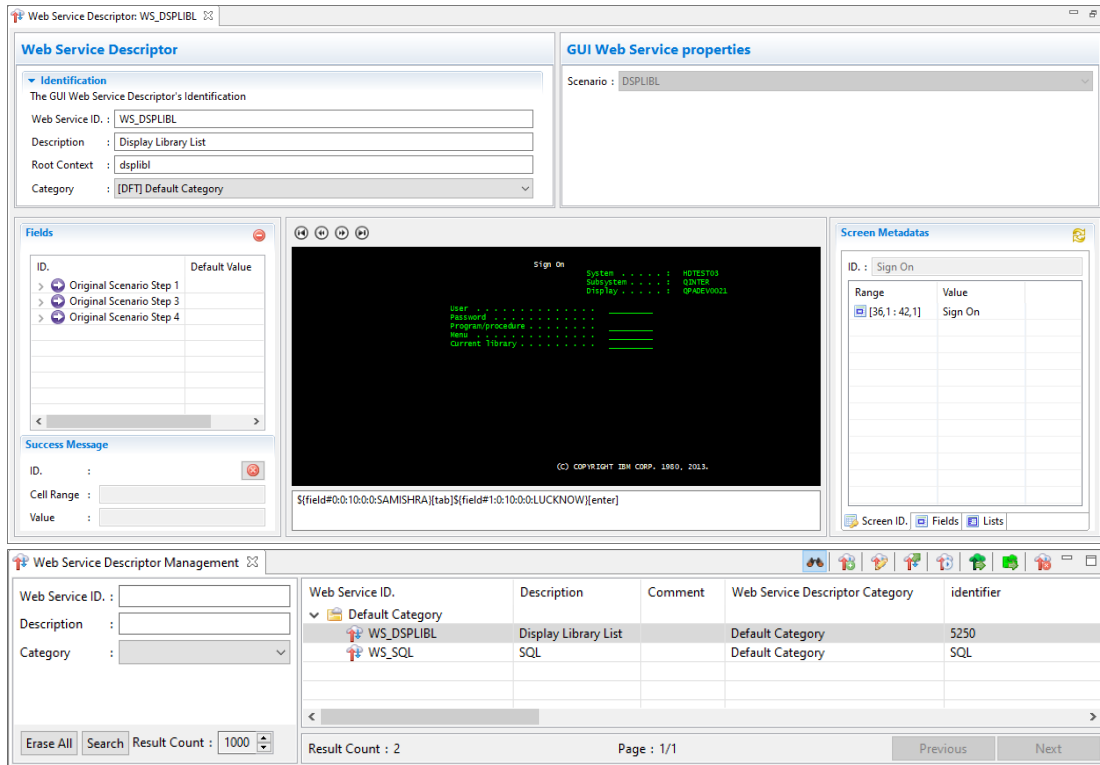


Figure 42: The web service descriptor editor

Web service descriptors are accessed and managed in the **Web Service Descriptors** view.

## 16.1 Creating web service descriptors

Follow the subsequent steps to create a new web service descriptor.

**Step 1** To access the **New Descriptor** wizard, click the Create icon in the toolbar of the **Web Service Descriptors** search view.

**Step 2** Select the **Web Service Type** in the drop-down menu.

- Select **5250** to create a web service based on a scenario.
- Select **SQL** to create a SQL web service.
- Select **REST** to turn a 5250 web service into a RESTful web service.

Click **Next >** to continue.

**Step 3** Define the descriptor's ID and enter a description. These values are required to create a new descriptor but can be edited later.

**Step 4** Define the **root context** to include in the web service's final URL. The text should be unique to this web service.

**Step 5** Select the **Category** in which to save the descriptor.


The list of categories includes all of the pre-defined categories available.

 **Reference**

For more information about creating and managing descriptor categories, refer to [Defining the default web service categories on page 56](#).

**Step 6** *[For 5252 web service types only]* Select the **Scenario** on which the 5250 web service descriptor will be based.

**Step 7** Click **Finish**.

**Result** The new web service descriptor is created and is displayed in the  **Web Service Descriptors** search view. The descriptor must be edited before it is ready for production.





## 16.2 Defining 5250 web service descriptors

---

Follow the subsequent steps to edit 5250 web service descriptors.

All changes made in the  **Web Service Descriptor** editor are saved automatically.

### 16.2.1 Editing 5250 descriptor details

To edit a descriptor's details, either right-click on it in the  **Web Service Descriptors** search view then select  **Edit properties**, or select the item then click the  Edit icon in the toolbar. You can also double-click on the  **Web Service Descriptor** to open its editor.

The descriptor's ID is displayed at the top of the editor.

#### Web Service ID

The descriptor's unique ID.

#### Description

A short description of the web service to help you identify what the scenario accomplishes.

#### Root context


This unique information will be used in the URL for the final web service.

#### Category

Select a category in which to organize the descriptor. The categories available in the drop-down list are those created in the studio preferences.

 **Reference**

For more information about creating categories, refer to [Defining the default web service categories on page 56](#).

The  **Web Service Descriptors** editor has a 5250 emulator that allows you to browse through all of the screens in the scenario. An image of the initial screen of each step in the scenario is displayed.

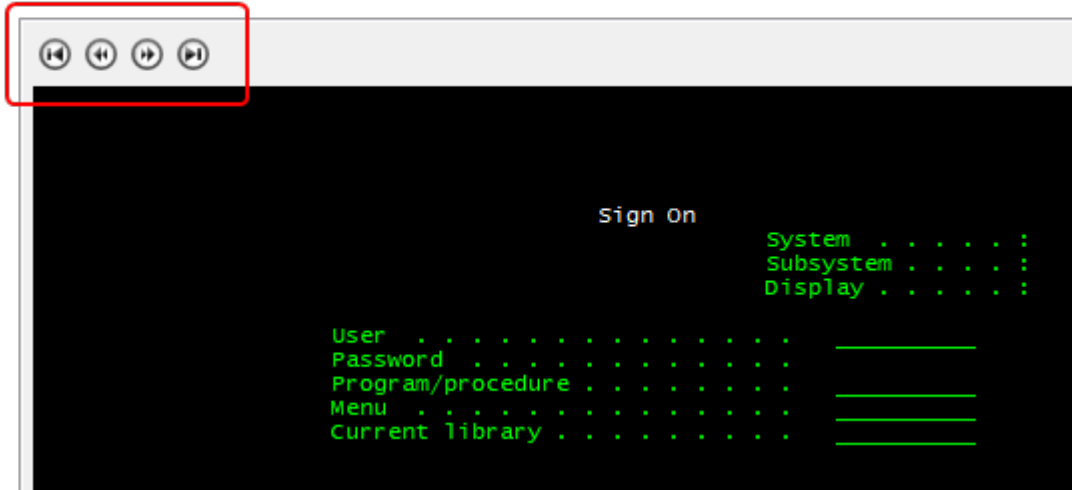






Figure 43: Navigate through the screens of the scenario to define the input/output values

Use the navigation buttons to browse through each screen.

-  navigate to the first step in the scenario
-  navigate to the previous step
-  navigate to the next step
-  navigate to last step in the scenario

The metadata for each screen are displayed in the **Metadata** section on the right.

## 16.2.2 Editing the 5250 web service properties

Define the execution properties for the 5250 web service.

### Scenario

Sets the scenario which the web service descriptor is based on. This field is not editable.

### Step Execution Response Time

Sets the delay between the scenario steps execution. The value is set in milliseconds and is the same for all of the scenario's steps.

### Global Timeout

Sets the global timeout for a web service execution. The web service execution aborts when the execution time exceeds this value. This value is set in milliseconds.

### Allowed Simultaneous executions

Sets the number of allowed executions on the web service descriptor.

## 16.2.3 Defining input and output content

On the left of the  **Descriptor** editor, in the **Fields** section, you can define the input and output values to use when replaying the scenario.

An input value type describes a field in which content/commands must be entered for the scenario to continue forward. Input data can only be a field ID, represented by '0'.

**Example**

Consider input fields as variables. The content of the input field is the value of the variable to apply to the field when the web service executes the scenario.

Output fields are fields whose content will be returned by the web service execution. Output data can be a field or a column ID, represented by '1'.

**Example**



Consider the output data as the result of an operation carried out during the scenario. The output the web service recovers may be the customer ID for a client. The scenario to execute is intended to recover the customer ID using a specific command. The value of the customer ID that is displayed is the output content that the web service is looking for; the result of the execution.

To tag fields/columns as places input/output data, each field/column must be identified via a metadata ID. Metadata must already be assigned to each field/column that is to be used as input/output sources.

**Reference**

For more information about creating metadata IDs, refer to [Metadata on page 73](#).

Follow the subsequent steps to tag fields/columns as input or output data.

**Step 1** Use the  forward and  back buttons to browse to a screen where a field or a column has been defined.

As you click through the screens, the **Metadata** section on the right displays the declared metadata for each screen so that you can easily see which screens require input.

**Step 2** Add the metadata in the **Fields** section.

- Drag each field from the **Screen Metadata** section and drop it to the **Fields** section on the left.
- You can also [ctrl+click] on several metadata to select them and drag and drop them.
- You can also double-click on the metadata to add it.

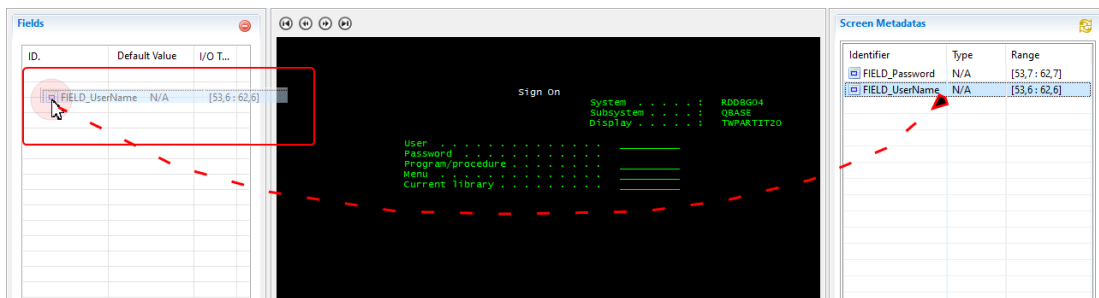


Figure 44: Drag a field ID into a descriptor to create input data

Metadata can only be used once in a descriptor. When you add metadata into the section, a dialog appears.




**Tip**

You can define multiple input/output fields at the same time: Ctrl+click] to select multiple fields and drag and drop them together into the **Fields** list on the left. All the fields can be defined in the dialog.

**Step 3** Define the properties for the added metadata in the dialog. If the metadata is a column, a field or a list, the properties to define are different. Refer to the [Field properties](#), [Column properties](#) or the [List properties](#) below.

**Step 4** Click **OK** to save.

**Result** The metadata value is defined for the web service as a child of the parent step in the **Fields** section.

## Field properties

### Field name

Sets the name of the field as defined in the **Metadata** editor. This value cannot be modified.

### Field Alias

Sets a **Field Alias** to attribute a unique identifier to the field. This option is useful if there are several input fields with the same name.

 **Note**

In the **Web Service Descriptor Execution** editor, the **Alias Name** is used instead of the **Field Name** in the **Input Fields List**.

### Data Type

Sets the data type of the field as defined in the Metadata editor. This value cannot be modified.

### I/O Type

Sets the type of value to create for the field: **input** or **output**.

### Default Value

For **input fields**, enter a **Default Value** to use automatically as the input value if the web service user does not assign a value for the field when replaying the scenario.

You can define an input field value as a reference to another input field value. Enter  *$\$refi:<another\_input\_field\_name>$*  in the **Default Value** field where *<another\_input\_field\_name>* is the name of the field from which you want to use the value.

You can define an external value as a reference to another input field value. Enter  *$\$refp:<properties\_name>$*  in the **Default Value** field where *<property\_name>* is the name of the property from which you want to use the value.

 **Note**

The reference fields tags ( *$\$refi\{\dots\}$* ,  *$\$refo\{\dots\}$* ,  *$\$refp\{\dots\}$* ) are case-sensitive. Enter them in lowercase.

**Note**

Field references support text decoration before and after the reference field tag, for example: ABC\${ref:<input\_field\_name>}XYZ.

### Mapping Table

For **output fields**, select a **Mapping Table** in the drop-down list to ensure that the value returned is translated with a specific chosen value. You can assign mapping entry to columns. The mapped value is applied to the output fields of the column.

### Required

Tick the **Required** box to make the input field mandatory. When executing the web services, an error is shown if the required fields are not defined.

### Column properties

#### Column name

The name of the column as defined in the Metadata editor. This value cannot be modified.

#### Data Type

The data type of the column as defined in the Metadata editor. This value cannot be modified.

### Mapping Table

Select a **Mapping Table** in the drop-down list to ensure that the value returned is translated with a specific chosen value. You can assign mapping entry to columns. The mapped value is applied to the output fields of the column.

### List properties

#### List Name

The name of the list as defined in the Metadata editor. This value cannot be modified.

#### Output Type

Select the output type of the list: **As Field** or **As Array**.

## 16.2.4 Defining success messages

You can define the conditions for what is considered a "functional success" of the execution of a web service. This means that the execution could be technically successful (the scenario has been successfully executed) but the functional execution could have failed.

**Example**

To identify a functional failure, you can use the content of a message that appears somewhere in a given step's screen.

To be able to use messages on screen as proof of functional success, each message must be identified via a metadata ID. A field ID must already be assigned to each message that is to be used as proof of success.

### Reference

For more information about creating metadata IDs, refer to [Metadata](#) on page 73.

Follow the subsequent steps to tag success messages.

- Step 1** Use the Forward and Back buttons to browse to a screen where a functional message has been defined.
- Step 2** Drag the field ID for the message from the **Screen Metadata** section and drop it to the **ID** field in the **Success Message** section on the left.

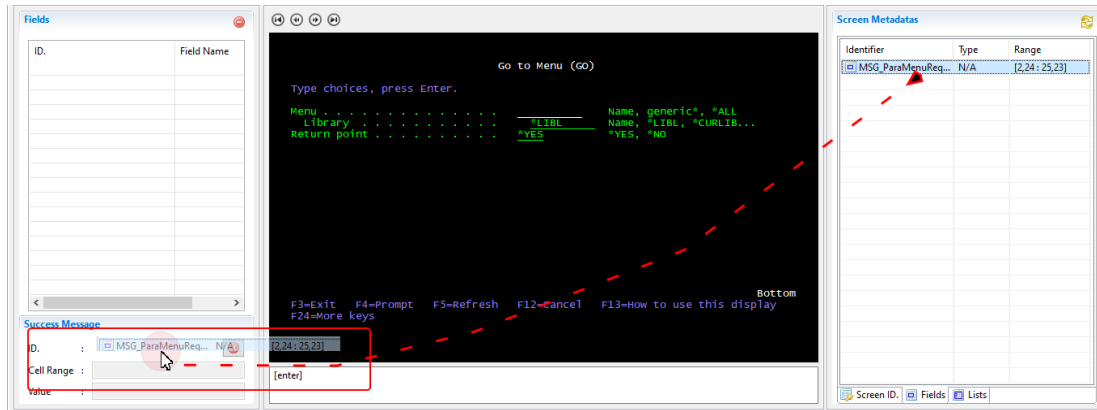


Figure 45: Drag a field ID into the Success Message ID field to create functional success cues

- Step 3** In the dialog, enter the **Expected Message**, word for word, that should appear when the execution is successful. It is important to enter the exact text of the message in order for the system to recognize it as proof of success.

You can reference the value of an input or an output field in the success message, using a variable. Enter `${refi:<field_name>}` in the **Default Value** field where `<field_name>` is the name of the input or output field from which you want to use the value.

### Example

Library `${refi:<input_field_name>}` is created.

**Result** The success message is defined for the whole scenario.

## 16.2.5 Defining a custom JSON output format for a web service

You can define a custom format template for the returned JSON file. The format editor is in the **Output Format** tab on the left section of the **Descriptor** editor. Enter the custom format template of the JSON output in the **Output Format** definition field.

To validate your template, click on the Validation icon. A dialog is displayed, showing if the output format is valid or invalid.

To display a simulation of the final JSON output result, click the Sample icon. ARCAD API generates fake values and uses the defined template to display them.

## Elements of customization

To define a custom Output Format, you have to define a valid JSON template including the following different elements:

- Field References
- List/Column References
- Execution Information References
- Static text

The available reference tags are:

Tag	Description
<code>&lt;field:[name_of_the_field]/&gt;</code>	Use to reference an output field value
<code>&lt;list:[name_of_the_list][display_type]&gt;</code>	Use to reference a list
<code>&lt;col:[name_of_column]/&gt;</code>	Use to reference a column
<code>&lt;instance/&gt;</code>	Use to reference the execution instance ID.
<code>&lt;messagecode/&gt;</code>	Use to reference the error code
<code>&lt;messagelvl1/&gt;</code>	Use to reference the error text level 1
<code>&lt;messagelvl2/&gt;</code>	Use to reference the error text level 2
<code>&lt;result/&gt;</code>	Use to reference the functional execution result
<code>&lt;status/&gt;</code>	Use to reference the execution result

Table 17: Reference tags to use in custom JSON output format

### Using a Field Reference Tag

To display a Field value , use the following Field Reference Tag in your template: `<field:[name_of_the_field]/>`

When generating the JSON output, this tag is replaced by the value of the Output Field

### Using a List Reference Tag

To display the values that come from a list, you must first use a List Reference Tag. This tag is used to define where the list is displayed and how the list content is displayed.

This tag is a Container Tag; this means that you also have to define child Column Reference Tags to used it.

To display a List values , use the following List Reference Tag in your template:

#### *List Reference Tag Syntax*

```
<list:[name_of_the_list][display_type]>
  <col:[name_of_col1]/>
  ...
  <col:[name_of_coln]/>
</list>
```

In this syntax sample,

- [name\_of\_the\_list] is the reference name of the list,
- [display\_type] defines the type of display (ARRAY\_OF\_OBJECTS or OBJECT\_of\_ARRAYS),
- [name\_of\_col] is the Reference Name of the column.

There are two types of display for list: ARRAY\_OF\_OBJECTS or OBJECT\_of\_ARRAYS.

#### Example

With the **ARRAY\_OF\_OBJECTS** type into the JSON Output Definition as follows:

*Array of objects list type*

```
<list:[name_of_the_list]|ARRAY_OF_OBJECTS>
  <col:[name_of_col1]/>
  ...
  <col:[name_of_coln]/>
</list>
```

You obtain the following:

The <list:....></list> tag will be replaced by the JSON string: []

The <col:[name\_of\_col1]/>,...,<col:[name\_of\_coln]/> tags will be replaced by the JSON string: {col1-value-1,..., coln-value-1},...,{coln-value-1,..., coln-value-m}

*Array of object list type JSON result*

```
[
  {
    col1-value-1
    ...
    coln-value-1
  },
  {
    col1-value-m
    ...
    coln-value-m
  }
]
```

#### Example

With the **OBJECT\_OF\_ARRAYS** type into the JSON Output Definition as follows:

*Objects of arrays list type*

```
<list:[name_of_the_list]|OBJECT_OF_ARRAYS>
```

```
<col:[name_of_col1]/>
...
<col:[name_of_coln]/>
</list>
```

You obtain the following:

The `<list:....></list>` tag will be replaced by the JSON string: `{}`

The `<col:[name_of_col1]/>` tag will be replaced by the JSON string: `["col1-value-1", ..., "col1-value-m"]`

*Objects of arrays list type JSON result*

```
{
  [
    col1-value-1,
    ...
    col1-value-m,
  ],
  [
    coln-value-1,
    ...
    coln-value-m,
  ]
}
```

 **Important!**


The usage of the list's **Display Type** is not enough to generate a valid custom JSON Output Format. Indeed, it is mandatory to setup the static text correctly to get a valid output.

 **Example**

Valid output format templates that generates valid custom JSON output.

*ARRAY\_OF\_OBJECTS Type*


```
{
  "MyList" : <list:[name_of_the_list]|OBJECT_OF_
ARRAYS>
    "mycol1" : <col:[name_of_col1]/>,
    "mycol2" : <col:[name_of_coln]/>
  </list>
}
```

 *OBJECT\_OF\_ARRAYS Type*

```


{
  "mylist" : <list:[name_of_the_list]|ARRAY_OF_
OBJECTS>
  "mycol1": "<col:[name_of_col1]/>",
  "mycol2": "<col:[name_of_coln]/>"
</list>
}

```

 **Important!**  
It is recommended to specify the type of value expected for each output in the custom JSON script.

The available values are the following:

- Alpha (string)
- Boolean
- Integer
- Float (decimal number)
- Date


 **Example**

```





{
  "Col1":<field:Col1|INTEGER/>,
  "Col2":<field:Col2|ALPHA/>,
  "Col3":<field:Col3|FLOAT/>,
  "Col4":<field:Col4|ALPHA/>
}

```

## 16.3 Defining SQL web service descriptors

Follow the subsequent steps to edit an SQL web service descriptor's details. All changes made in the  **Web Service Descriptor** editor are saved automatically.

### 16.3.1 Editing SQL descriptor details

To edit a descriptor's details, either right-click on it in the  **Web Service Descriptors** search view then select  **Edit properties**, or select the item then click the  Edit icon in the toolbar. You can also double-click on the  **Web Service Descriptor** to open its editor.

The descriptor's ID is displayed at the top of the editor.

### Web Service ID

The descriptor's unique ID.

### Description

A short description of the web service to help you identify what the scenario accomplishes.

### Root context

This unique information will be used in the URL for the final web service.

### Category

Select a category in which to organize the descriptor. The categories available in the drop-down list are those created in the studio preferences.



#### Reference

For more information about creating categories, refer to [Defining the default web service categories on page 56](#).

## 16.3.2 Defining an SQL Select Query

In the middle of the editor, you can write **SQL** logic to provide values for **SQL** based web services. The input and output fields are included within the **SQL** logic.

#### Note

Define the **Input Fields** and **Output Fields** in order to test the execution of the web service.

The SQL query uses an SQL inherited syntax and allows you to define the Input and Output fields.

The syntax is the following one:

```
SELECT <output_field_definition1>, <output_field_definition2>, ..., <output_
field_definitionN>
FROM <from_clause>
WHERE <where_clause_including_input_field_definitions>
```

### Defining output fields

The Output fields are the fields that are returned into the JSON Content of the Web Service.

To define the output field, use the following syntax:

```
${o:<output_fieldname>:<column_name>}
```

where:

- <output\_fieldname> is the name used into the JSON Output Result,
- <column\_name> is the name of the column of the table you want to get the data.



## Defining input fields

To define the input field, use the following syntax:

```
${i:<input_fieldname>:<datatype_id>}
```

where:

- <input\_fieldname> is the name used to define input data,
- <datatype\_id> identifies the datatype. Use **0** for Integer, **1** for Float, **2** for String.

The output field definitions will appear into the select clause and the input field definition generally appear into the where clause.

### Example

```
SELECT ${o:CODE:APP_CAPP}, ${o:TEXT:APP_CTXT}
FROM ARCAD_PRD/AARCAPPF1
WHERE APP_CAPP like '${i:APPLICATION_CODE:2}%'
```

In this query:

The values of the columns *APP\_CAPP* and *APP\_CTXT* will be extracted from the table *ARCAD\_PRD/AARCAPPF1* and will be respectively returned as a value of the *CODE* and *TEXT* of the JSON Result.

The table will be filtered to the content of the *APP\_CAPP* column and will return only the records where the value of the *APP\_CAPP* column starts with a input value defined by the user. The API call must include a input parameter called *APPLICATION\_CODE*.

### 16.3.3 The SQL execution process

For SQL APIs, the execution process:


1. Discovers all the output field definitions.
2. Extracts the related column names.
3. Replaces the output field definition by their related <column\_name>.
4. Discovers all the input field definitions.
5. Replace the input field definition by the input value defined by the user.
6. Execute the SQL Query.
7. Get the value of all the columns defined into the output field definition.
8. Generate the JSON Result.

### Example

Assign *BUI* to the input field *APPLICATION\_CODE*.

1. Discovers all the output field definitions:

```
${o:CODE:APP_CAPP}
```



`#{o:TEXT:APP_CTXT}`


2. Extracts the related column names:  
`APP_CAPPAPP_CTXT`
3. Replaces the output field definition by there related `<column_name>`:  





```
SELECT APP_CAPP, APP_CTXT
FROM ARCAD_PRD/AARCAPPF1
WHERE APP_CAPP like '#{i:APPLICATION_CODE:2}%'
```
4. Discovers all the input field definitions:  
`#{i:APPLICATION_CODE:2}`
5. Replace the input field definition by the input value defined by the user:  

```
SELECT APP_CAPP, APP_CTXT
FROM ARCAD_PRD/AARCAPPF1
WHERE APP_CAPP like 'BUI%'
```
6. Execute the SQL Query.
7. Generate the JSON Result:  

```
{
  "CODE": "BUILDER ",
  "TEXT": "ARCAD-Builder Demo Application"
},
{
  "CODE": "BUILDTTEST ",
  "TEXT": "buildttest "
}
```

## 16.4 Defining REST web service descriptors

Follow the subsequent steps to edit a REST web service descriptor's details. All changes made in the  **Web Service Descriptor** editor are saved automatically.

To edit a descriptor's details, either right-click on it in the  **Web Service Descriptors** search view then select  **Edit properties**, or select the item then click the  Edit icon in the toolbar. You can also double-click on the  **Web Service Descriptor** to open its editor.

The descriptor's ID is displayed at the top of the editor.

### Web Service ID

The descriptor's unique ID.

### Description

A short description of the web service to help you identify what the scenario accomplishes.

### Root context

This unique information will be used in the URL for the final web service.

### Category

Select a category in which to organize the descriptor. The categories available in the drop-down list are those created in the studio preferences.



#### Reference

For more information about creating categories, refer to [Defining the default web service categories on page 56](#).

## 16.4.1 Creating REST actions

**Step 1** Click the  Create Rest action icon in the **Rest Actions** section.

**Step 2** Select the **HTTP Method** in the dialog.

**Step 3** Enter the **URI** of the REST action. The variable segment must be put in brackets.

**Step 4** Click the  Browse button to select the **Linked Descriptor**. Select the web service descriptor that will be executed for the selected HTTP method. Click **OK**.

**Step 5** Click **OK**.

**Result** The REST action is created and is displayed in the **REST Actions** section. The parameters values that have been extracted from the input URI are displayed in the **Parameters** section.

## 16.4.2 Mapping a Variable Segment to an Input Parameter

The list of **Parameters** contains the parameter for each variable segment that has been extracted when defining the REST action. You need to map each of the parameter to an input field.

**Step 1** Double-click a parameter to open the **Parameter Definition** dialog.

**Step 2** Select the input field you want to map (the input field that will receive the value of the Variable Segment during the execution) in the **Input Field Identifier** drop-down list. The list contains the input fields defined for the related web service descriptor.

**Step 3** Click **OK**.



#### Important!

All parameters must be mapped.



## 16.5 Testing the execution of the web service

This feature allows you to test the execution of the web service. Testing your web services is an important step in creating APIs. You can catch any anomalies, clean up the process and find out what can be improved before it becomes available.

There are two ways of testing the execution of a web service. You can run an execution of the service as a whole, or execute it step by step.

### 16.5.1 Execute a test of a web service

Follow the subsequent steps to run a test execution of a web service.

**Step 1** Right click on web service descriptor in the  **Web Service Descriptors** view and select  **Execute (test)** in the contextual menu to open the **Test Execution** editor.

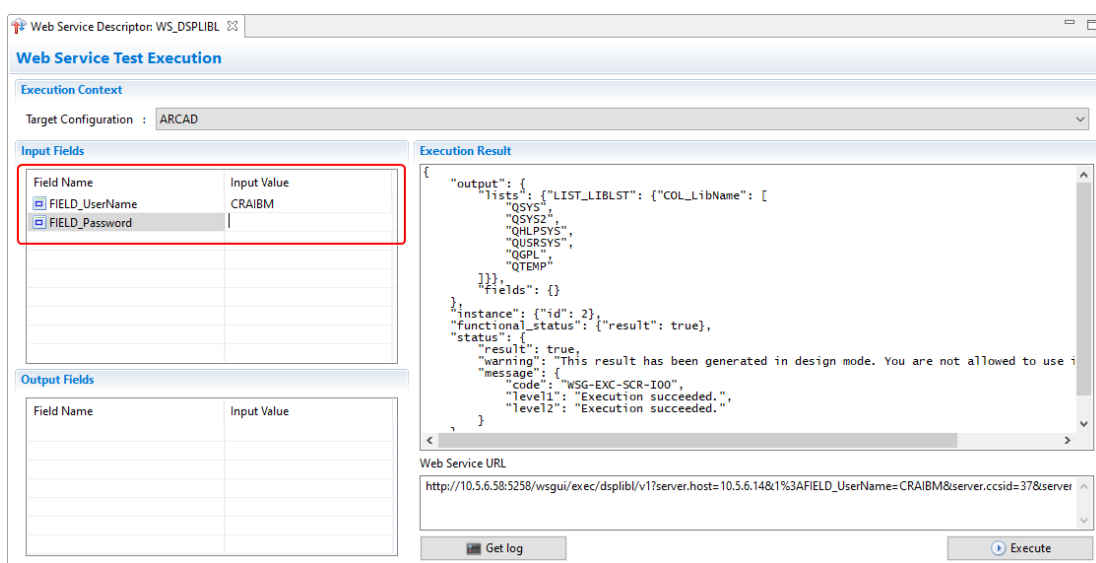


Figure 46: Web service test

**Step 2** Select a **Target Configuration** to define the execution context. This is the machine on which the execution will be run. The list is prepopulated with the list of sessions available.



#### Important!

Define the correct **Login** and **Password** of the target configuration in order to execute a SQL query-based web service.

**Step 3** [For REST web service descriptors] Select the rest action to test on the REST Actions drop-down list.

**Step 4** Define the **Input Fields** and **Output Fields** values that are needed to execute the web service. For SQL web services, the fields are retrieved automatically from the SQL query.

The **Web Service URL** is displayed with the input and output field once these fields are defined.

**Step 5** Click the  **Execute** button.

**Result** The test execution is launched and when it is finished, the execution results are displayed in the **Execution Results** section.

## 16.5.2 Execute a test of a web service step by step

Executing the web service in this way can be very useful to debug step by step. Follow the subsequent steps to run a test execution of a web service step by step.

**Step 1** Right click on web service descriptor in the **Web Service Descriptors** view and select **Execute (test)** in the contextual menu to open the **Test Execution** editor.

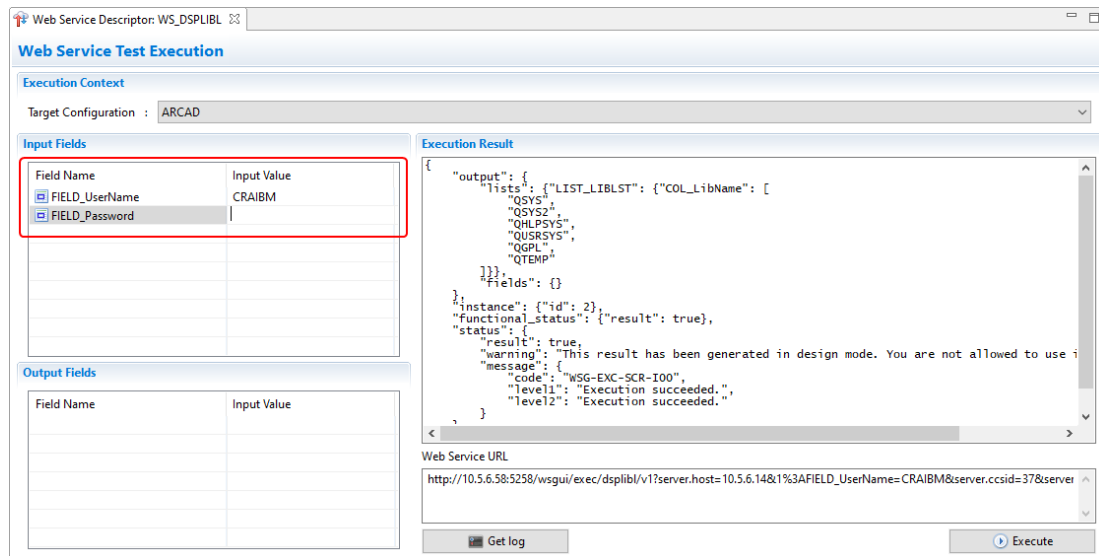


Figure 47: Web service test

**Step 2** Select a **Target Configuration** to define the execution context. This is the machine on which the execution will be run. The list is prepopulated with the list of sessions available.

**Important!**  
Define the correct **Login** and **Password** of the target configuration in order to execute a SQL query-based web service.

**Step 3** [For REST web service descriptors] Select the rest action to test on the REST Actions dropdown list.

**Step 4** Define the **Input Fields** and **Output Fields** values that are needed to execute the web service. For SQL web services, the fields are retrieved automatically from the SQL query.

The **Web Service URL** is displayed with the input and output field once these fields are defined.

**Step 5** Click the **Execute** button.

**Result** The execution starts and then stop after the end of the first step of the scenario is reached.

**Step 6** Click the Next icon to execute the next step, and so on until you reach the end of the execution.

**Result** The test execution is launched and when it is finished, the execution results are displayed in the **Execution Results** section.

### 16.5.3 View execution results

**Execution Results** are in JSON format. They display output and field data, and other information concerning the execution of the web service.

 **Note**

Web services using a Success Message return an Error message in the JSON execution result if the captured value does not match with the one that has been defined as a Success Message value.

Click the  **Get log** button to download a log file. For a step by step execution, they are available once the execution is complete.

For 5250 descriptors, the log file shows the different screens that have been used during the execution of the web service. With this log, any unexpected screens used can be easily identified and the scenario can be modified accordingly.

For all descriptor types, the time consumption during the test execution a web service is stored at the end of the log file. The values given are in milliseconds.

 **Important!**

These values are stored in the log file, do the acces to this file must be possible. The returned content in the JSON file must include the instance ID. So, if you use HTTP headers to return values or if you use a customized JSON output format without including this value, you will not be able to access to the instance ID.

 **Reference**

For more information about HTTP Headers, refer to [Defining the use of HTTP headers on page 50](#).

For more information about customized JSON output format, refer to [Defining a custom JSON output format for a web service on page 99](#).

The related URL to call the web service is displayed in the **Web Service URL** field. This URL is updated each time a value changes in the editor (input values or target configuration).

Copy the URL and paste it into a browser. A REST request is executed and the tool replies in a REST format (JSON) containing the expected result. This result can be read by a web application and handled by web developers.

## 16.6 Exporting web service descriptors

Export a descriptor from ARCAD-API to share it and make it available to be imported into a different ARCAD-API production server other than the local one accessible from your current ARCAD-API Studio.

Exporting a descriptor will create a compressed file that contains all of the elements required to run the web service on a production server.

For **5250** web services:

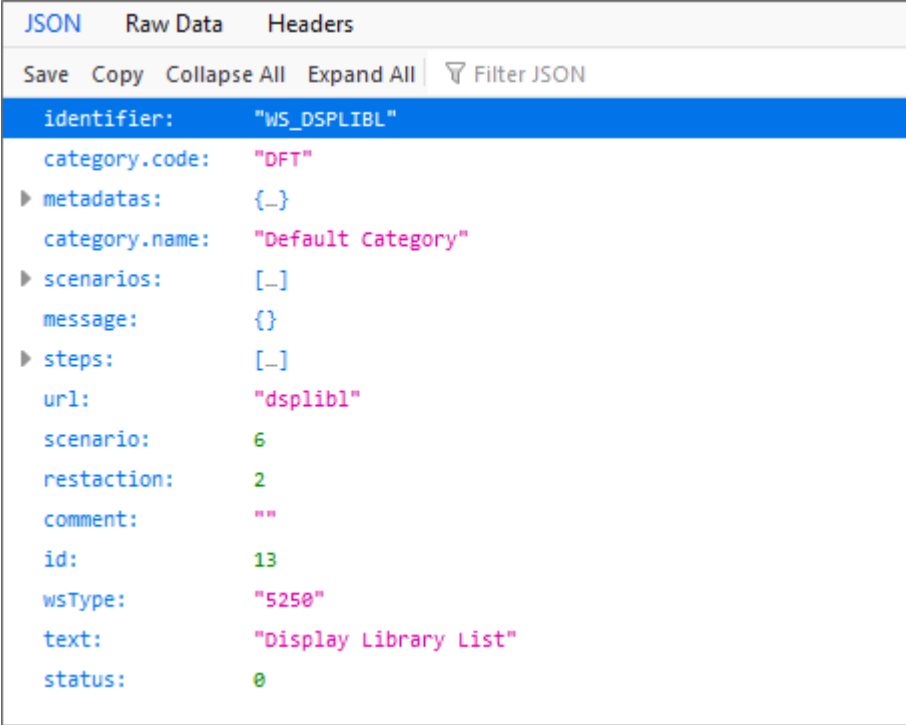
- The web service type: **5250**,
- The web service descriptor's properties (identifier, category code, and category name, etc.),
- The scenario's properties,
- The metadata used by the descriptor or by one of the involved scenarios,
- The steps containing screen content captured during the scenario recording.

For **SQL** web services:

- The web service type: **SQL**,
- The web service descriptor's properties (identifier, category code, and category name, etc.),
- The *Query* information.

For **REST** web services:

- The web service type: **REST**,
- The web service descriptor's properties (identifier, category code, and category name, etc.),
- The information about the 5250 linked scenario,
- The information about the rest actions and their parameters.





```

JSON  Raw Data  Headers
Save Copy Collapse All Expand All Filter JSON
identifier: "WS_DSPLIBL"
category.code: "DFT"
▶ metadatas: {}
category.name: "Default Category"
▶ scenarios: []
message: {}
▶ steps: []
url: "dsplibl"
scenario: 6
restaction: 2
comment: ""
id: 13
wsType: "5250"
text: "Display Library List"
status: 0
  
```

Figure 48: Contents of the export JSON file

Follow the subsequent steps to export a descriptor to run your web service outside of the ARCAD-API Studio.

- Step 1** Right click on a web service descriptor in the list and select  **Export** or select the descriptor and click the  Export icon.
- Step 2** Browse to select the desired location in which to save the file in the **Save As** dialog and enter a name for the exported file.
- Step 3** Click the **Save** button to export the package.

**Result** The exported package can now be imported into an ARCAD-API production server.

 **Reference**

For more information about importing a descriptor into a production server, refer to [Importing web service descriptors on page 131](#).

## 16.7 Exporting web service descriptors' Open API specifications

---

Generate a documentation for the web service descriptor compliant to the Open API specification. You can choose to use the Swagger 2.0 version or the Open API 3.0 version of the standard.


 **Reference**

For more information about Open API standard, refer to:

<https://spec.openapis.org/oas/v3.1.0> or  
<https://swagger.io/resources/open-api/>

Exporting a descriptor will create a compressed file that contains all of the elements required to describe the web service according to the Open API specification.

Follow the subsequent steps to export a descriptor specification to the open API standard.

**Step 1** Right click on a web service descriptor in the list and select  **Export Specification** or select the descriptor and click the  Export Specification icon.

**Step 2** Define the specification version: select **2.0** to use Swagger 2.0 or **3.0** to use Open API 3.0 in the **Open API Version** drop-down list.

**Step 3** Browse to select the desired location in which to save the file in the **Save As** dialog and enter a name for the exported file.

**Step 4** Click the **Save** button to export.

**Result** The Open API web service specification is exported to the JSON format.

Use the following web service URL for displaying the documentation:

`http://<server>:<port>/arcapi/export/specification/<specVersion>/<webServiceDescriptorCode>`


## 16.8 Promoting descriptors to the production server

---

Once a web service has been designed and tested, it must be available for the final API consumer. To make it available, you must promote it into production. Promote a descriptor to push it to the internal production server available by default in ARCAD-API.


Every time you promote the same web service descriptor to production, a new version is added to the list. Newer versions of the web service don't override the previous versions.


Follow the subsequent steps to promote a web service descriptor to the production server.

**Step 1** Right click on a web service in the list and select  **Promote** or select the descriptor and click the  Promote icon.




**Step 2** Select the **Production Package** in which to save the descriptor from the drop-down list. This list is pre-populated with all of the production packages available on the local production server.

 **Important!**  
In order to save a web service in a package on the production server, the package must already exist.

 **Reference**  
For more information about creating production packages, refer to [Production packages on page 121](#).

**Step 3** Enter a **Version Description** to summarize the details of the current version of the descriptor that is being pushed to the production server.

**Step 4** Click **Finish**.

**Result** The promoted web service can be found in the  **Registered Web Services** view. After promoting your web service you must also create the packaged version before it is available to be consumed.

 **Reference**  
For more information about accessing promoted web services, refer to [Registered web services on page 131](#)

## 16.9 Promoting entire descriptor categories to the production server

---

Promote entire categories to push them to the internal production server available by default in ARCAD-API. When a category is pushed to production, all of the web service descriptors in the category are pushed at the same time.

Follow the subsequent steps to promote a web service descriptor category to the production server.

**Step 1** Right click on category in the list and select  **Promote category** or select the category and click the  Promote Category icon.

**Step 2** Select the **Production Category** in which to save the descriptors from the drop-down list. This list is pre-populated with all of the production packages available on the local production server.

 **Reference**  
For more information about production categories, refer to [Production packages on page 121](#).

**Step 3** Enter a **Version Description** to summarize this delivery.

**Step 4** Click **Finish**.

**Result** The promoted web services can be found in the  **Registered Web Services** view.




 **Reference**  
For more information about accessing promoted web services, refer to [Registered web services on page 131](#)

## 16.10 Deleting web service descriptors

---

 **Warning!**

Deleted web service descriptors cannot be accessed or recovered.

To delete a web service descriptor, either right-click on the item in the  **Web Service Descriptors** view and select  **Delete**, or select the item and click the  Delete icon in the toolbar. Click **OK** to confirm or click **Cancel** to keep the web service descriptor.



# HOSTING WEB SERVICES




# Introduction to hosting web services

---

ARCAD-API comes with an embedded production server. This server can be used to host your APIs. This section describes what this production server is intended to bring to your APIs and how to use the different tools available in it.


- [IBM i Production server settings on page 117](#)
- [Production packages on page 121](#)
- [Version tags on page 124](#)
- [Registered web services on page 131](#)
- [API Gateways settings on page 133](#)

# 17 IBM i Production server settings

Access  ARCAD-API Server →  Embedded Production Server →  IBM i Production Server Settings

The web services that have been registered on the production server can only be executed by one dedicated IBM i server. This means that the consumer of the web services cannot choose which target IBM i to use. The imposed target that will run the web service is defined in the production server settings.

**Note**  
The ARCAD-API license management is based on this server.

The production server's information is accessed and managed in the  **IBM i Production Server Settings** view. Open this editor to:

- define the 5250 connection parameters that will be used to execute the web services,
- register or check the license key,
- define contact information for the exposed API.

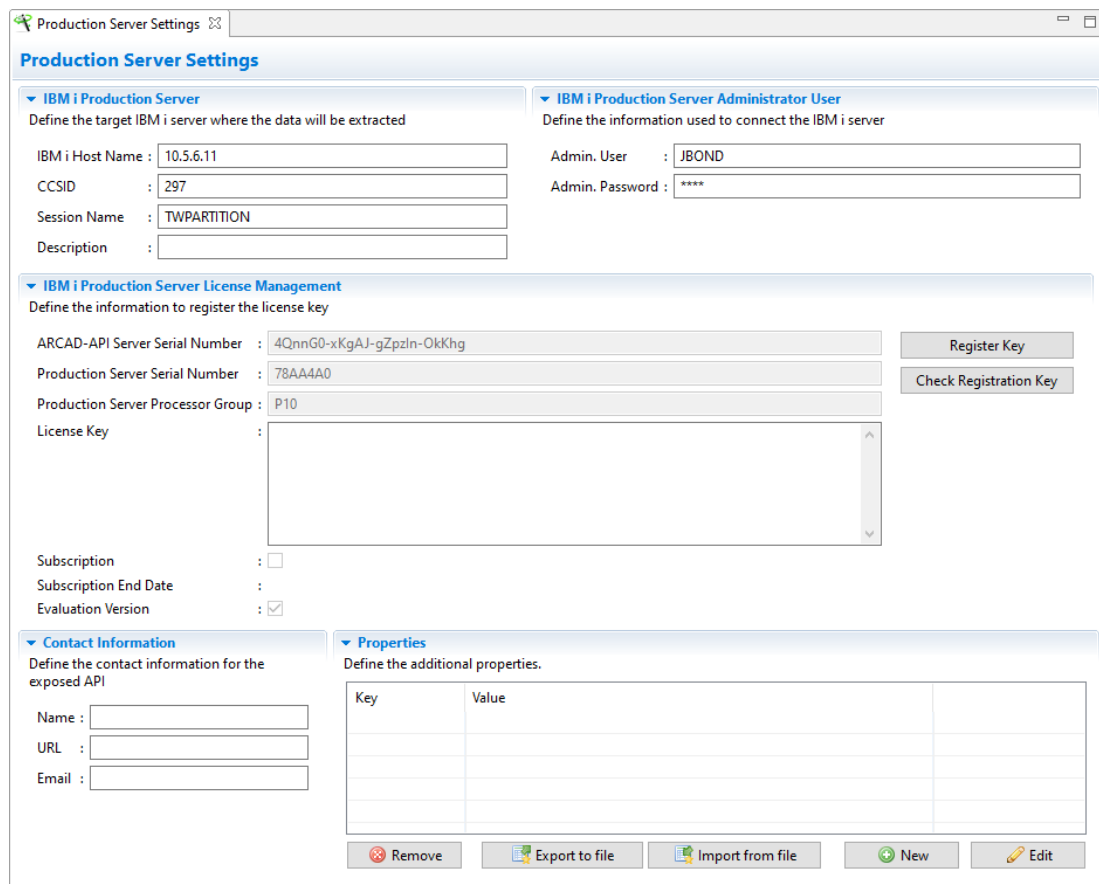


Figure 49: The IBM i Production Server Settings view

## IBM i Host Name

The IP Address or the DNS of the IBM i Production Server.

## CCSID

The CCSID used to open the 5250 session.

### Session Name

Enter a name for the server to identify it.

### Description

Enter a short description for the server for more identification information.

### Admin User & Admin Password

The user and password to use to access the server.

The first time you install the ARCAD-API Server, an evaluation license key is automatically generated and is available for 60 days. You don't need to enter any license key to activate it. To check how many days are remaining on your evaluation license, click the **Check Registration Key** button.

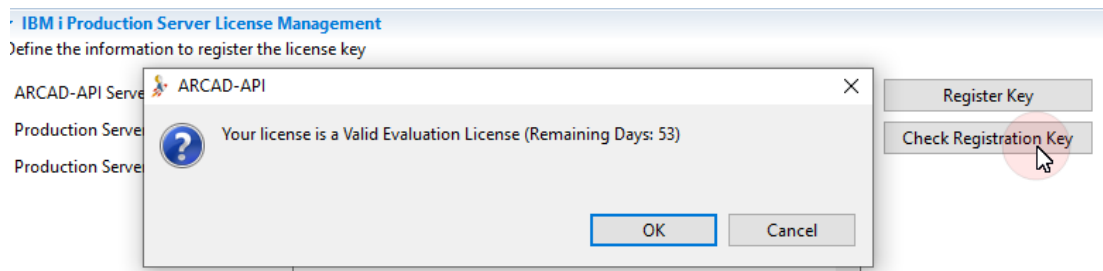


Figure 50: Check how many days remain on your evaluation license

To get a permanent license key, you must provide some information to the ARCAD Software Support Service and they will calculate an activation key. Please send the following information:

- The ARCAD-API Server Serial Number
- The IBM i Production Server Serial Number
- The IBM i Production Server Processor Group

### ARCAD-API Server Serial Number

This is the serial number for your installation of ARCAD-API. Send this number to ARCAD to receive a permanent (paid) license for this tool.

### Production Server Serial Number & Production Server Processor Group

These fields will be automatically populated when the user/password to access the server are entered and the server settings are saved. Send this information to ARCAD to receive a permanent (paid) license for this tool.

### License Key

Copy your permanent (paid) license key into this area, then click the **Register Key** button to activate it.

**Warning!**  
If you changed the IBM i Production Server properties after having registered the key, the server will no longer work.

**Note**  
Once a license key is validated, it is automatically put into use and you will no longer be using the evaluation mode.

### Subscription

According to your contract, you may be able to use ARCAD-API under a subscription license. This means that you can use all the features for a limited time. At the end of this period you must renew your license. Registering a Subscription License Key uses the same process used to register a Permanent License Key.

If you have registered a subscription license key the **Subscription** checkbox is checked.

### Subscription End Date

The limit date on which your subscription license will expire.

### Evaluation Version




If your server is in evaluation mode, this checkbox is checked automatically.

You may want to define contact information for the exposed API. These information are needed for Swagger/Open API documentation.

- The identifying **Name** of the contact person/organization.
- The **URL** pointing to the contact information. The format must be an URL.
- The **Email** address of the contact. person/organization. The format must be an email address.

You may want to define properties for the production server.




Properties are pairs of keys and values that together create variables which can be used when defining actions inside deployment processes. If you enter a property's key in a field when defining an action, the value defined in the same property will be used. Properties refer to specific entity-related content and can only be referenced if the entity in which they are defined is used in the current deployment process. They are intended to help reuse details concerning your entities.

- To access and manage the properties defined in an entity, select the  **Properties** tab in the entity's editor.
- To create a new property manually, click the  **Create** button. Both the **Key** and **Value** fields are mandatory.
- If the property is a password, check the **Is Password** checkbox. When the property is saved, the **Value** will be hidden and display as asterisks (\*).
- To create multiple properties automatically for most entities, click the  **Import from File** button to and select a predefined `.properties` file to upload. The properties to import must follow this syntax: `key = value`.

#### Note

The `.properties` file can contain more information than the values to import but only the content in the `key = value` syntax will be imported into the list of properties.

The imported properties are only available for the current entity. To reuse imported properties in other entities, import the file again in the second entity's editor.

- To export the properties defined for any entity as a `.properties` file, click the  **Export to file** button. All of the properties defined in the tab are exported.
- To edit an existing property, double-click it, or select it, then click the  **Edit** button.
- To delete a property, select it, then click the  **Delete** button.




 **Warning!**

Deleted properties are no longer associated with any variable and cannot be accessed or recovered.

Save the changes (, Ctrl+S or **File > Save**).



# 18 Production packages

Access  ARCAD-API Server →  Embedded Production Server →  Production Packages

## Chapter Summary

18.1 Creating production packages .....	122
18.2 Editing production packages .....	123
18.3 Accessing version tags .....	123

Production packages are sets of promoted web services. They may include a set of web services that are to be managed together as a group or may represent the same 5250 application and provide APIs that cover a coherent set of features. Because you may use ARCAD-API to provide web services for all kinds of miscellaneous internal systems and scenarios, the notion of production packages may help the consumers to clearly choose which services to use.

Another benefit of packaging multiple web services into production sets is the ability to then provide versions. One or multiples web services could be modified according to the evolution of the target system and, in this case, all the related web services must be updated. The package can be redelivered with updated web services under a new version number. In ARCAD-API, the version of a production package is called the version tag.

To avoid forcing the final consumer to keep track of each version of each individual web service used, the standard way of managing such a situation is to create API versions instead of web service versions. In this case, the consumer must only keep track of the version tag of the API ensemble and the URL prefix to use. If a new version of the API is available because one or more web services have been modified, the consumer only needs to change the version tag without needing to know the exact version of each web service.

 **Important!**

Production packages cannot be deleted.

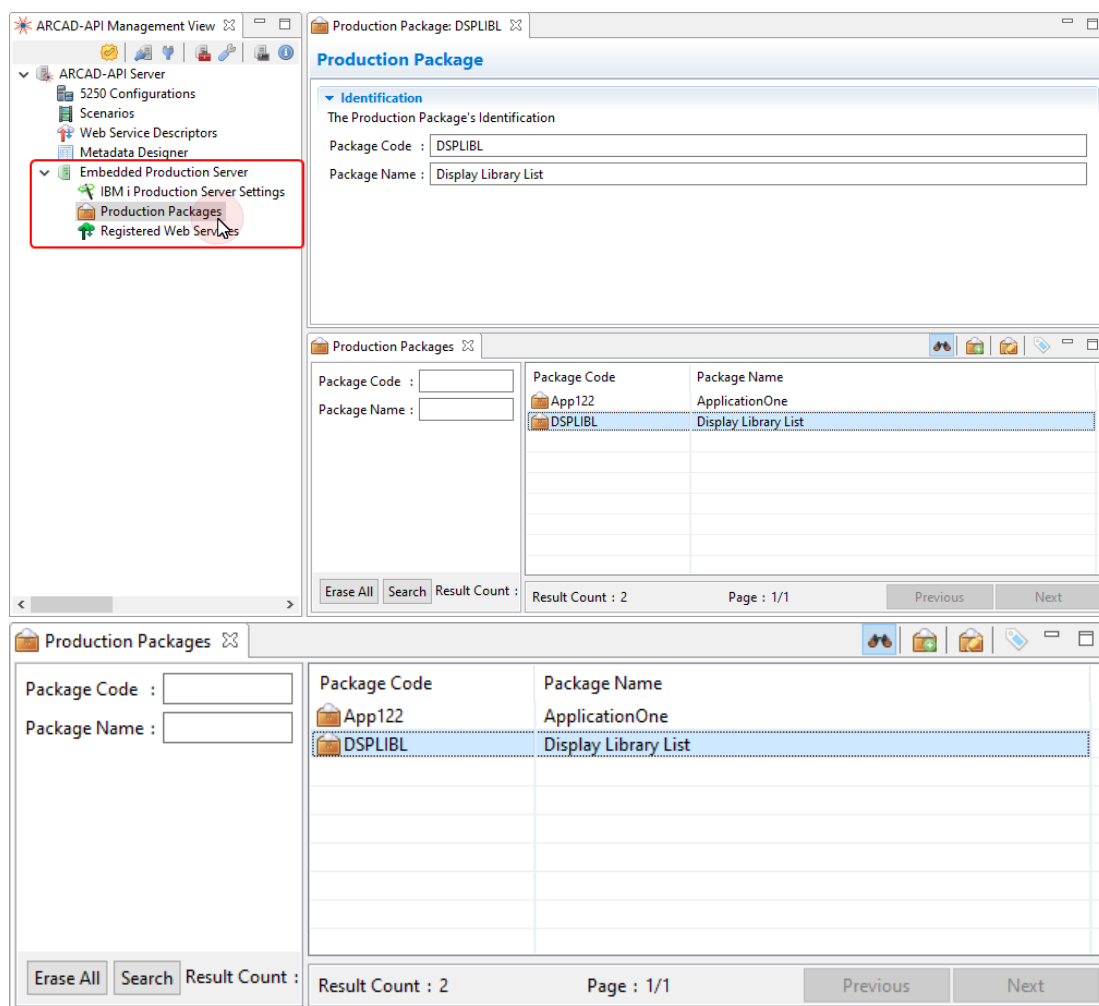





Figure 51: Production packages in the production server

Production packages are accessed and managed in the  **Production Packages** view.

## 18.1 Creating production packages

Follow the subsequent steps to create a new production package.

- Step 1** To access the **New Production Package** wizard, either click the  Create icon in the toolbar, or right-click anywhere in the list and select  Create.
- Step 2** Define the code and name for the package. These values are required to create a new package but can be edited later.
- Step 3** Click **Finish**.
- Result** The new package is created and is displayed in the  **Production packages** search view. The package can now be used to accept internal promotions.

### Reference

For more information about promoting web services to production, refer to [Promoting descriptors to the production server on page 112](#) and for their use in production, refer to [Registered web services on page 131](#).

## 18.2 Editing production packages

---

To edit a production package's details, either right-click on it in the  **Production packages** search view then select  **Edit**, select the item then click the  edit icon in the toolbar or double-click it.

Only the packages **Code** and **Name** can be modified in the editor.

Save the changes (, Ctrl+S or **File > Save**).

## 18.3 Accessing version tags

---





An important feature for production packages is the capability to manage the version of the set of web services. The latest version of all the web services that belong to a package will be included in each defined version tag. Only version tags are available for consumption. If you have not created a version tag for a production package, the registered web services in it will not be available.

To display a production package's version tag, right-click on the production package in the list and select  **Show version tags** in the contextual menu.

 **Reference**

For more information about version tags, refer to [Version tags on page 124](#).

# 19 Version tags

Access  ARCAD-API Server →  Embedded Production Server →  Production Package →  Version tags


## Chapter Summary

19.1 Accessing version tags.....	124
19.2 Creating version tags.....	125
19.3 Editing version tags.....	125
19.4 Promoting versions to the Kong server.....	126
19.5 Registering versions in APIGEE Edge.....	129
19.6 Exporting web services' Open API specifications.....	130

Version tags are used to create versions of web service packages and therefore make them available to be consumed.

An important feature for production packages is the capability to manage the version of the set of web services. ARCAD-API offers the capability to define version tags that group multiple versions of web services that belong to the same production package under a unique version number. Even if each web service in the set is currently at a different version level, the final user will only have access to the versions included in the package's version tag. A version tag is a list of multiple web services (in the package) that each have different version levels themselves.

While creating a version tag for a category, ARCAD-API will include the latest version of all the web services that belong to the package.

 **Important!**  
Only tagged versions of production packages can be promoted to an external Kong server to delegate the management.

As soon as a version tag has been created and activated, the related API is available for the consumer. Web services are available at the following URL:

`https://<arcad-api-server>:<arcad-api-port>/arcapi/exec/<package-code>/<version-tag-number>/<web-service-root-context>`

## 19.1 Accessing version tags

**Step 1** Open the **Production Packages** view and select the production package.

**Step 2** Right-click on the package to display the contextual menu and select the **Show Version Tags** option.

Version tags are accessed and managed in the  **Version Tags** search view.

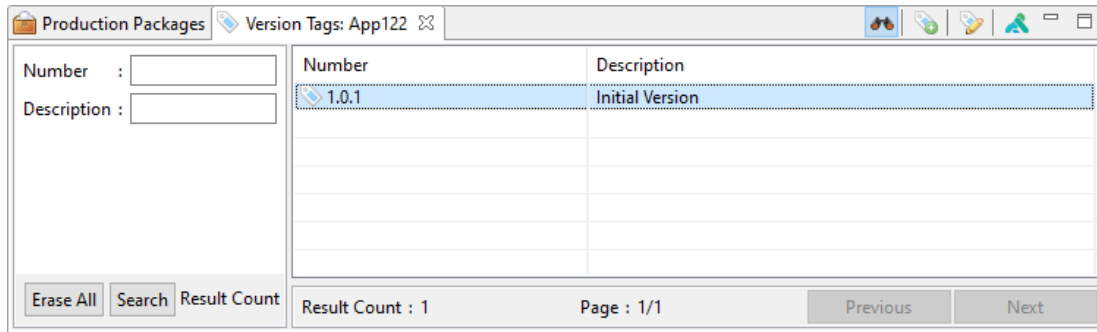


Figure 52: The Version Tags search view

To refine the list of existing version tags displayed, use the Search tool.



- To search for an existing version tags by number or description, enter all or part of the number or description in the corresponding field.

Enter any combination of the above search criteria, then click the **Search** button to display the results. To display the complete list, click the **Search** button without entering any search criteria.

To display all of the items by default each time the search view is opened, select the  auto search icon.


## 19.2 Creating version tags

Follow the subsequent steps to create a new version tag.

**Step 1** To access the **New Version Tags** wizard, either click the  Create icon in the toolbar of the **Version Tags** search view, or right-click anywhere in the view and select  **Create**.

**Step 2** Define the version tag's **Number** and **Description**. These values are required to create a new tag but can be edited later.

**Step 3** Click **Finish**.



**Result** The new version tag is created and is displayed in the  **Version Tags** search view. Because the version tag is automatically associated with the parent production package, the most recent version of each promoted web service is automatically added to the version.

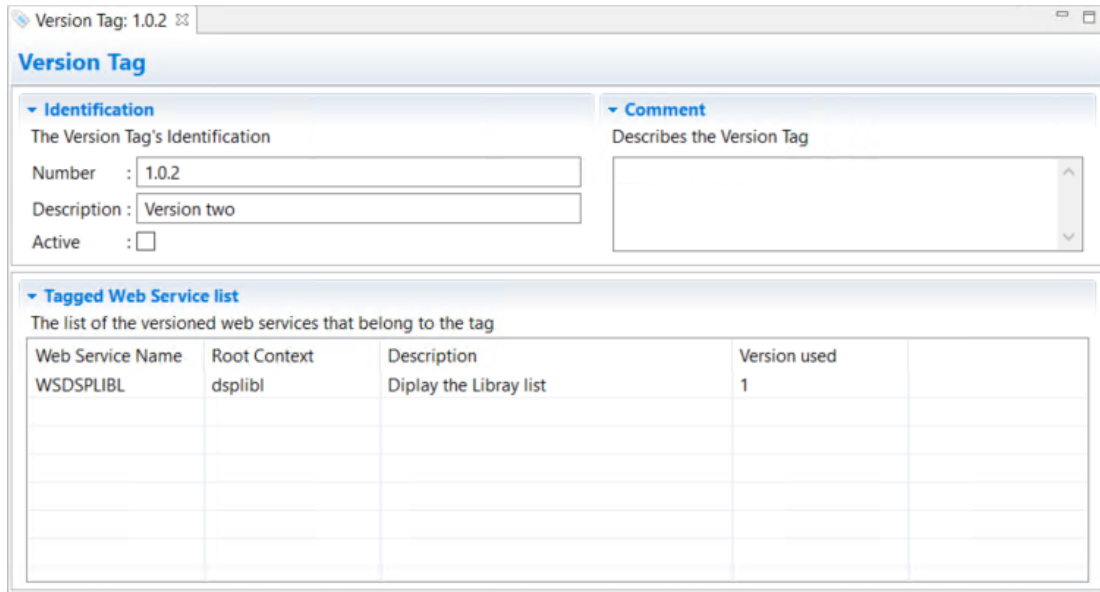
*Once it is activated, the version will be able to be called by external consumers.*

### Reference

For more information about the URLs to use to access your APIs, refer to the [Web services catalog on page 140](#).

## 19.3 Editing version tags

To edit a version tag's details, either right-click on it in the  **Version Tags** search view then select **Edit**, select the item then click the  edit icon in the toolbar or double-click it.



Web Service Name	Root Context	Description	Version used
WSDSPLIBL	dsplibl	Diplay the Libray list	1

Figure 53: The Version Tag editor

### Number

The version number.

### Description

The short description of the version.

### Active

Tick this checkbox to make the version tag available for consumption.

**Important!** This option must be activated to make your web services available.

### Comment

A detailed description of the version.

### Tagged web services

The list of web services included in the version. The most recent version of each web service in the production package is automatically included in the version tag.

Save the changes (, Ctrl+S or **File > Save**).

## 19.4 Promoting versions to the Kong server

 Kong is an Open Source API Gateway ([konghq.com](https://konghq.com)) that provides the following features:

- Authentication
- Traffic Control
- Analytics
- Transformations
- Logging

ARCAD-API delegates API management to the external gateway Kong. In order to use your API, you must register it with Kong and set up the required elements. ARCAD-API enables you to register sets of APIs together via the version tag. Registering a tag with Kong will register all of the web services available in it and they will all become available via a single service.

#### Important!

ARCAD-API only registers a Kong Service and the required elements to make it work. The API management strategy belongs to the Kong Administrator so this means that you cannot create other management tools, like Route for example, from ARCAD-API.

#### Reference

To communicate with the Kong server, the HTTP request needs to use a Basic Authentication. This means that the Kong Server has to be aware of that authentication to be able to proxy the request to the ARCAD-API Server.

To manage this connection information, refer to [Kong Gateway on page 137](#).

Follow the subsequent steps to register a service for each version of a production package.

**Step 4** From the  **Version Tags** search view, select the tag to register a service.

**Step 5** Right-click on it to display the contextual menu and select **Register in Kong**.

**Step 6** Once the registration is finished, the following data is returned by the Kong Admin API.

Kong Service List

```
GET http://<kong_server_address>:<admin_port>/services
```

```

1 | {
2 |   "next": null,
3 |   "data": [
4 |     {
5 |       "host": "xxx.xxx.xx.x",           <- The ARCAD-
API Server IP Address or DNS
6 |       "created_at": 1584625667,
7 |       "connect_timeout": 60000,
8 |       "id": "62b9a612-8089-47cb-b083-bd2140fdbf0e",
9 |       "protocol": "http",
10 |      "name": "TestPromoteV1",
11 |      "read_timeout": 60000,
12 |      "port": 6200,                       <- The ARCAD-
API Server Port
13 |      "path": "/arcapi/exec/.../V1",      <- The API
Execution Path for the registered Version Tag
14 |      "updated_at": 1584625667,
15 |      "retries": 5,
16 |      "write_timeout": 60000,
17 |      "tags": null,
18 |      "client_certificate": null
19 |    }
20 |  ]
21 | }
```

**Note**

The protocol, host and port attributes are extracted from the Base URL defined in the server configuration. See: [Kong Gateway on page 137](#)

The path attribute is the root URI used to execute a web service exposed in the related version tag.

The name attribute is assigned using the concatenation of the parent production package code and the version number of the version tag.

## Service Plugin registration

```
GET http://<kong_server_address>:<admin_port>/services/<version>/plugins
```

```
22 | {
23 |   "next": null,
24 |   "data": [
25 |     {
26 |       "created_at": 1584625667,
27 |       "config": {
28 |         "remove": {
29 |           "querystring": [],
30 |           "headers": [],
31 |           "body": []
32 |         },
33 |         "replace": {
34 |           "querystring": [],
35 |           "headers": [],
36 |           "uri": null,
37 |           "body": []
38 |         },
39 |         "http_method": null,
40 |         "add": {
41 |           "querystring": [],
42 |           "headers": [
43 |             "Authorization: Basic
YWRtaW5AcXVhZHJhOnF1YWRYQ==" <- The Authorization key
44 |           ],
45 |           "body": []
46 |         },
47 |         "append": {
48 |           "querystring": [],
49 |           "headers": [],
50 |           "body": []
51 |         },
52 |         "rename": {
53 |           "querystring": [],
54 |           "headers": [],
55 |           "body": []
56 |         }
57 |       },
58 |       "id": "445f7a06-8d6a-459e-b289-89ea6ade1f08",
59 |       "service": {
60 |         "id": "62b9a612-8089-47cb-b083-bd2140fdbf0e"
61 |       },

```



```

GET http://<kong_server_address>:<admin_port>/services/<version>/plugins
62     "enabled": true,
63     "protocols": [
64         "grpc",
65         "grpc",
66         "http",
67         "https"
68     ],
69     "name": "request-transformer", <- The request-transformer
plugin installed
70     "consumer": null,
71     "route": null,
72     "tags": null
73   }
74 ]
75 }


```

### Note

The `data/config/add/headers[0]` is an header that defines the Basic Authentication String used by Kong to connect to the ARCAD-API Server to execute the REST API.

This connection string is built using the ARCAD-API user defined in the server configuration. See: [Kong Gateway on page 137](#).



## 19.5 Registering versions in APIGEE Edge

ARCAD-API delegates API management to the external gateway  APIGEE Edge. In order to use your API, you must register it with APIGEE and set up the required elements. ARCAD-API enables you to register sets of APIs together via the version tag. Registering a tag with APIGEE will register all of the web services available in it and they will all become available via a single service. You can also deploy the web-services tagged to APIGEE Edge just after registering.

The registration process is the following one:

- Copy and unzip the selected template zip file.
- For each web service that belongs to the selected version tag:
  - Substitute the variables in to template xml file.
  - Zip the modified directory.
  - Call the related APIGEE Rest API to create a KVM that will contains the ARCAD-API Credentials.
  - Call the related APIGEE Rest API to register the new API Proxy.
  - If registration is OK, retrieve the revision number.
  - If the **Deploy after registering** option has been checked, all the related APIGEE Rest API to deploy the newly revision into the selected target environment.

Follow the subsequent steps to register a version tag to an APIGEE API Proxy.

**Step 1** Select the version tag you want to register, click the  Register icon in the toolbar of the **Version Tags** search view, or right-click on the Version Tag in the view and select  Register.

**Step 2** In the APIGEE API Proxies Registration wizard, select the **APIGEE Registration Template** you want to use in the drop-down list.

**Step 3** [*Optional*] Check the **Deploy after registering** box.

If you select the **Deploy after registering** option, the web-services tagged to the selected version are deployed on the selected Target Environment once the registration is completed.


**Step 4** Select the APIGEE Environment to deploy to in the Target Environment drop-down list.

**Step 5** Click **Finish**.

**Result** The registration is done.

## 19.6 Exporting web services' Open API specifications

---

Generate a documentation for the web services compliant to the  Open API specification. You can choose to use the Swagger 2.0 version or the Open API 3.0 version of the standard.



### Reference

For more information about Open API standard, refer to:

<https://spec.openapis.org/oas/v3.1.0> or  
<https://swagger.io/resources/open-api/>

Exporting a descriptor will create a compressed file that contains all of the elements required to describe the web service according to the Open API specification.

Follow the subsequent steps to export a web service specification to the open API standard.

**Step 1** Right-click on a version tag in the list and select  **Export Specification** or select the version tag and click the  Export Specification icon.

**Step 2** Define the specification version: select **2.0** to use Swagger 2.0 or **3.0** to use Open API 3.0 in the **Open API Version** drop-down list.

**Step 3** Select the web services you wish to export. Click to select the web service in the list, or tick the **Select All** box to include all web services of the package's version.

**Step 4** Browse to select the desired location in which to save the file in the **Save As** dialog and enter a name for the exported file.

**Step 5** Click the **Save** button to export.

**Result** The Open API web service specification is exported to the JSON format.

Use the following web service URL for displaying the documentation:

`http://<server>:<port>/arcapi/export/specification/<specVersion>/<packageCode>/<package_version>/<webServiceDescriptorCode>`

## 20 Registered web services

Access ARCAD-API Server → Embedded Production Server → Registered Web Services

### Chapter Summary

20.1 Importing web service descriptors .....	131
20.2 Demoting registered web services .....	132

Promoted web services are available for consumption via the embedded ARCAD-API production server.

Web services are either automatically made available in the **Registered Web Services** view because they were promoted there via the local designer or because they were imported into the production server manually.

When new versions of web service descriptors are promoted to the production server, they are added to the list of versions available in the registered web services' editor. By default, the most recent version of each web service listed is included in the version tag of a production package.

#### Reference

For more information about pushing web services to the ARCAD-API production server, refer to:

[Promoting descriptors to the production server on page 112](#)

[Promoting entire descriptor categories to the production server on page 113](#)

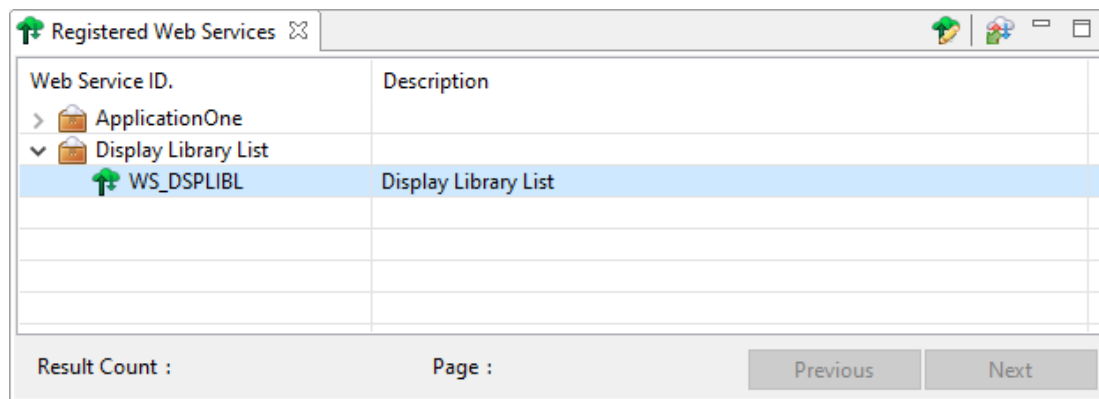


Figure 54: The Registered Web Services view

Registered web services are accessed and managed in the **Registered web services** view. Click on any registered web service to access the different versions of the descriptor that have been promoted to the production server.

### 20.1 Importing web service descriptors

Import a web service into your local ARCAD-API production server to recover web services created on a different ARCAD-API designer server.

 **Note**

Importing web services manually into the local production server is an alternative to promoting them to the local production server. It is recommended that you import only when sharing web services between different designer servers. If you need to access a web service via a different production server, outside of the imbedded production server, you can export the descriptor instead of promoting it.

The automatic promotion to local production feature is available to avoid this extra hassle.

 **Reference**




For more information about exporting web service descriptors, refer to [Exporting web service descriptors on page 110](#).

When a web service descriptor package is imported, its content is analyzed and extracted starting at the root context of the descriptor. If needed, a new web service descriptor category is created on the production server to accommodate the hierarchy of the original descriptor. The next version number is generated according to the existing content. A new web service descriptor version directory is created including a *version.properties* file.

 **Note**

By default, the version description is set to *Imported*.

Follow the subsequent steps to import a web service descriptor package.


- Step 1** To open the import wizard, either right-click anywhere in the  **Registered Web Services** view and click  **Import**, or click the  Import icon in the toolbar.
- Step 2** Select the **Production Package** in which to save the descriptor from the drop-down list. This list is pre-populated with all of the production packages available on the local production server.

 **Important!**

In order to save a web service in a package on the production server, the package must already exist.

 **Reference**



For more information about creating production packages, refer to [Production packages on page 121](#).

- Step 3** Browse to the exported package *.zip* file.
- Step 4** Click the **Finish** button to import the package.
- Result** The imported package is now available on the ARCAD-API production server from the  **Registered Web Services** view.

## 20.2 Demoting registered web services

---

Web service descriptors can be unregistered from the production server.

Right-click on a registered web service in the list and select  **Unregister** or select the descriptor and click the  Unregister icon. Click **OK** to confirm or click **Cancel** to keep the web service.

# 21 API Gateways settings

Access  ARCAD-API Server →  Embedded Production Server → API Gateway Settings

## Chapter Summary

21.1 APIGEE Gateway.....	133
21.2 WSO2 Settings.....	137
21.3 Kong Gateway.....	137

ARCAD-API is compatible with any API gateway, however some configuration is required to use some of them. In the production server, you can set up and configure the connections to the API gateways.


### Reference

The ARCAD-API Server's default configuration for API gateways are defined in the its configuration view. Refer to:

[Defining the APIGEE Edge Settings on page 48,](#)

[Defining the API Gateway General Settings on page 49.](#)

## 21.1 APIGEE Gateway

ARCAD-API provides the necessary mechanisms to register a Web Service in  Google APIGEE as an API Proxy.

### 21.1.1 APIGEE Settings

These settings allows the gateway administrator to define the information required:

- to connect to the APIGEE server,
- to define the target organization,
- to define some technical information related to the registration of an API Proxy.

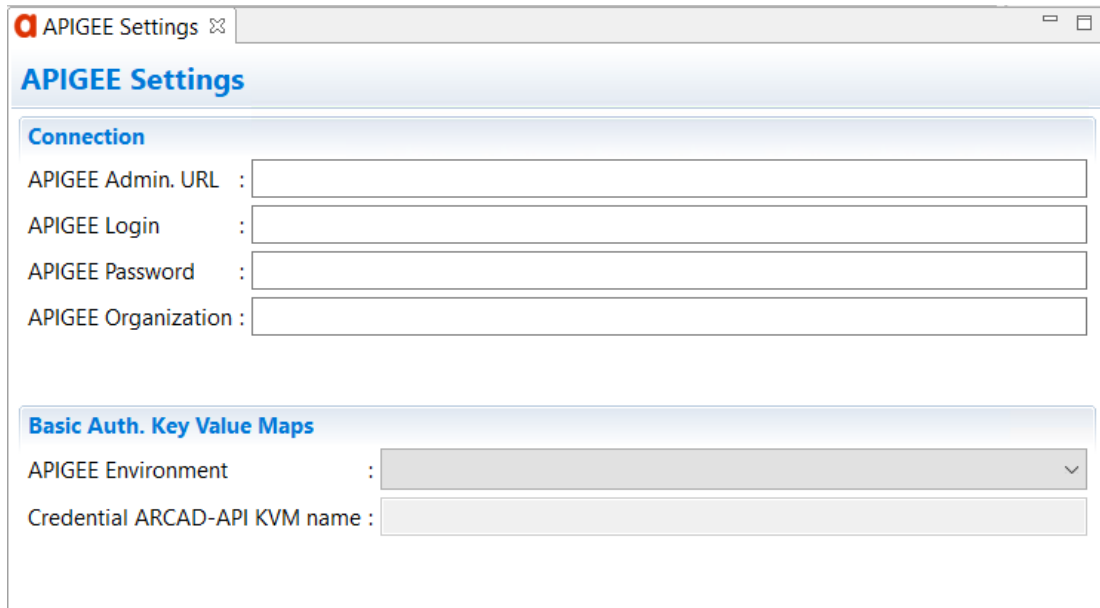


Figure 55: Define the APIGEE settings

### APIGEE Admin. URL

The URL of the APIGEE server.

### APIGEE Login and APIGEE Password

The user login and password to connect to the APIGEE server.

### APIGEE Organization

The name of the target organization.

### APIGEE Environment

The name of the environment on which the ARCAD API credential will be created.

### Credential ARCAD API KVM name

The name of the Key Value Map used to register the ARCAD API credential.

Save the changes (, Ctrl+S or **File > Save**).

## 21.1.2 APIGEE Templates

The registration of an APIGEE API Proxy is done by sending a zip file to the APIGEE server. This zip file contains a directory where all the required definition *.xml* files are stored.

The definition of an APIGEE API Proxy could contain a lot of sub-definitions to match the Enterprise policies. Because ARCAD API is not a APIGEE server front-end (this means that it does not provide a Graphical User Interface to APIGEE), it is not possible to define all these parameters at ARCAD API level.




To solve this point, ARCAD API offers the capability to import APIGEE API Proxy Declaration Templates that will be used during the registration process.

A template is a zip file that contains all the required definition of an APIGEE API Proxy in which we will substitute some value, before registering an API Proxy, to make it point to the right web service.

### 21.1.2.1 Creating APIGEE API Proxy templates

**Step 1** To access the  **New APIGEE Templates** wizard, either click the  Create icon in the toolbar, or right-click anywhere in the list and select  Create.


**Step 2** Define the **Name** and **Description** for the template. These values are required to create a new template but can be edited later.

 **Note**  
 Editing a template allows you to only change the name or the description of the template. If you want to change the template content, you need to use the  Update Template option.


**Step 3** In the **File Name** field, click the  Browse icon to select an API Proxy zip file.



**Step 4** Check the **Variabilize the package before import** box to add automatically the substitution variables at the right place.

**Step 5** Click **Finish** to start the import of the information from the file.

**Result** When the import is finished, the new template is created and is displayed in the  **APIGEE Templates** search view. The template can now be used to define an APIGEE API Proxy.

### 21.1.2.2 Updating APIGEE API Proxy templates

 **Important!**  
 When you update an APIGEE template, all the configuration defined previously for the template will be lost.

**Step 1** To update a template, either right-click on it in the  **APIGEE Templates** search view then select  **Update**, select the item then click the Update icon in the toolbar.

**Step 2** In the **File Name** field, click the  Browse icon to select a new API Proxy zip file.

**Step 3** Check the **Variabilize the package before import** box to add automatically the substitution variables at the right place.

**Step 4** Click **Finish** to start the import of the information from the file.

**Result** When the import is finished, the template is updated and is displayed in the  **APIGEE Templates** search view.

### 21.1.2.3 Package variabilization Process

For the package variabilization process, some substitution variables are added in the following files:

- aproxy.xml
- targets/default.xml

#### On the *apiproxy.xml* file

The /APIProxy/Basepaths text is replaced by:

```
/${arcadapi.versiontag}/${arcadapi.rootcontext}
```

**On the *targets/default.xml* file**

The TargetEndpoint/HTTPTargetConnection/URL text is replaced by:

```
${arcadapi.baseurl}${arcadapi.execroute}/${arcadapi.packagecode}/${arcadapi.versiontag}/${arcadapi.rootcontext}
```

The TargetEndpoint/HTTPTargetConnection/URL/Path text is replaced by:

```
${arcadapi.execroute}/${arcadapi.packagecode}/${arcadapi.versiontag}/${arcadapi.rootcontext}
```

**⚠ Important!**

The name of the proxy descriptor file must absolutely be ***apiproxy.xml***. It means that you need to change its name manually if you decide to import a template based on a downloaded API Proxy revision because in that case, the name of the descriptor file is by default the name of the related API proxy.

**21.1.2.4 Substitution variables**

Substitution Variable	Value
<code>\${arcadapi.versiontag}</code>	The Version Number of the selected version tag
<code>\${arcadapi.rootcontext}</code>	The Web Service of the selected version tag
<code>\${arcadapi.baseurl}</code>	The public URL of the ARCAD-API Server
<code>\${arcadapi.execroute}</code>	/arcapi/exec
<code>\${arcadapi.packagecode}</code>	The code of the related Production Package

Table 18: Substitution Variable

**⚠ Important!**

The value used to replace the variable `${arcadapi.baseurl}` comes from the field ARCAD-API Base URL of the ARCAD-API - Commons Settings/API Gateway General Settings server settings.


The values used to create the KVM ARCAD-API Credential comes from the fields ARCAD-API Use\*r and \*ARCAD-API Password of the ARCAD-API - Commons Settings/API Gateway General Settings server settings.

**📖 Reference**

Refer to [Defining the API Gateway General Settings on page 49](#).



## 21.2 WSO2 Settings

The web services provided by ARCAD API can be invoked from a  WSO2 gateway using JSON Web Token-based authentication.

The WSO2 API Gateway can be used to call ARCAD API Web Service using a JWT as an authentication mechanism. To validate a JWT, the WSO2 Public Key used to encode the token is required.

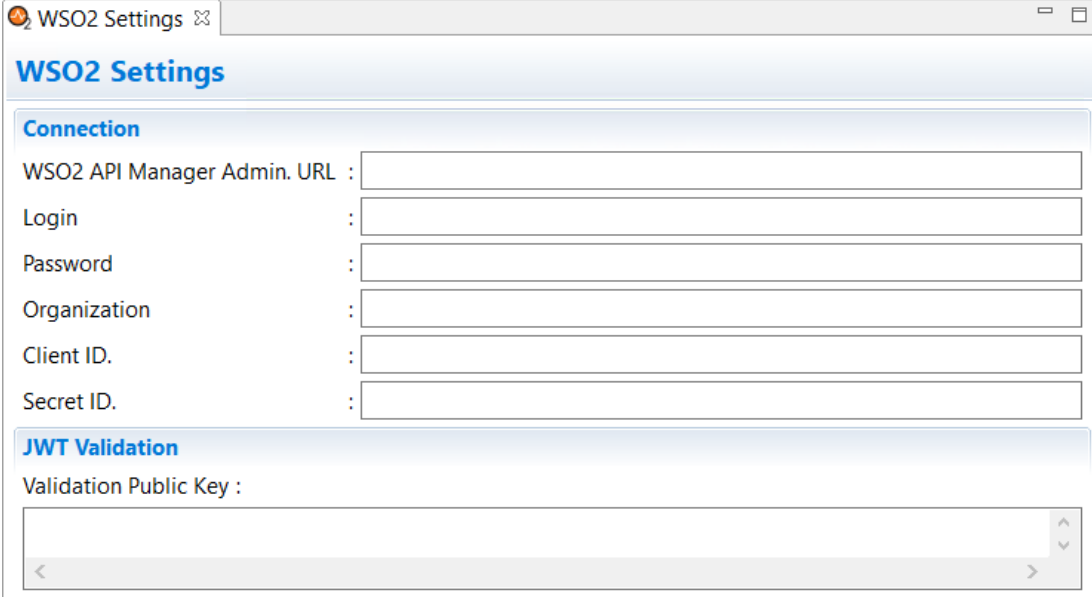


Figure 56: Define the WSO2 settings

### WSO2 API Manager Admin. URL

The URL of the WSO2 API Manager server.

### Login and Password

The user login and password to connect to the WSO2 API Manager server.

### Organization

The name of the target organization.

### Client ID and Secret ID

The Client ID and the Client Secret ID of the target organization.

### JWT Validation - Validation public key

Enter the WSO2 Public Key in to the **Validation Public Key** field of the JWT Validation section. To find the WSO2 Public Key, you need to export it from the wso2carbon.jks file.

Save the changes (, Ctrl+S or **File > Save**).

## 21.3 Kong Gateway

Define the  Kong gateway connection properties.

### Kong API Administration URL

Enter the Admin URL to connect to the Kong Server.

`http://<kong_server_address>:<admin_port>`

Generally the port used is 8001.



*Figure 57: Define the Kong settings*

 **Reference**

For more information about registering your API(s) on the Kong server, refer to [Promoting versions to the Kong server on page 126](#).



# APPENDICES

## Web services catalog

---

An online catalog of your registered web services can be accessed to display the documentation of your registered web services, one specific web service or a specific version of a web service.

The documentation is available via the following URLs:

- <http://<server>:<port>/arcapi/catalog>
- <http://<server>:<port>/arcapi/help/all>
- [http://<server>:<port>/arcapi/help/<web-service\\_root\\_context>](http://<server>:<port>/arcapi/help/<web-service_root_context>)
- [http://<server>:<port>/arcapi/help/<web-service\\_root\\_context>/<version\\_number>](http://<server>:<port>/arcapi/help/<web-service_root_context>/<version_number>)

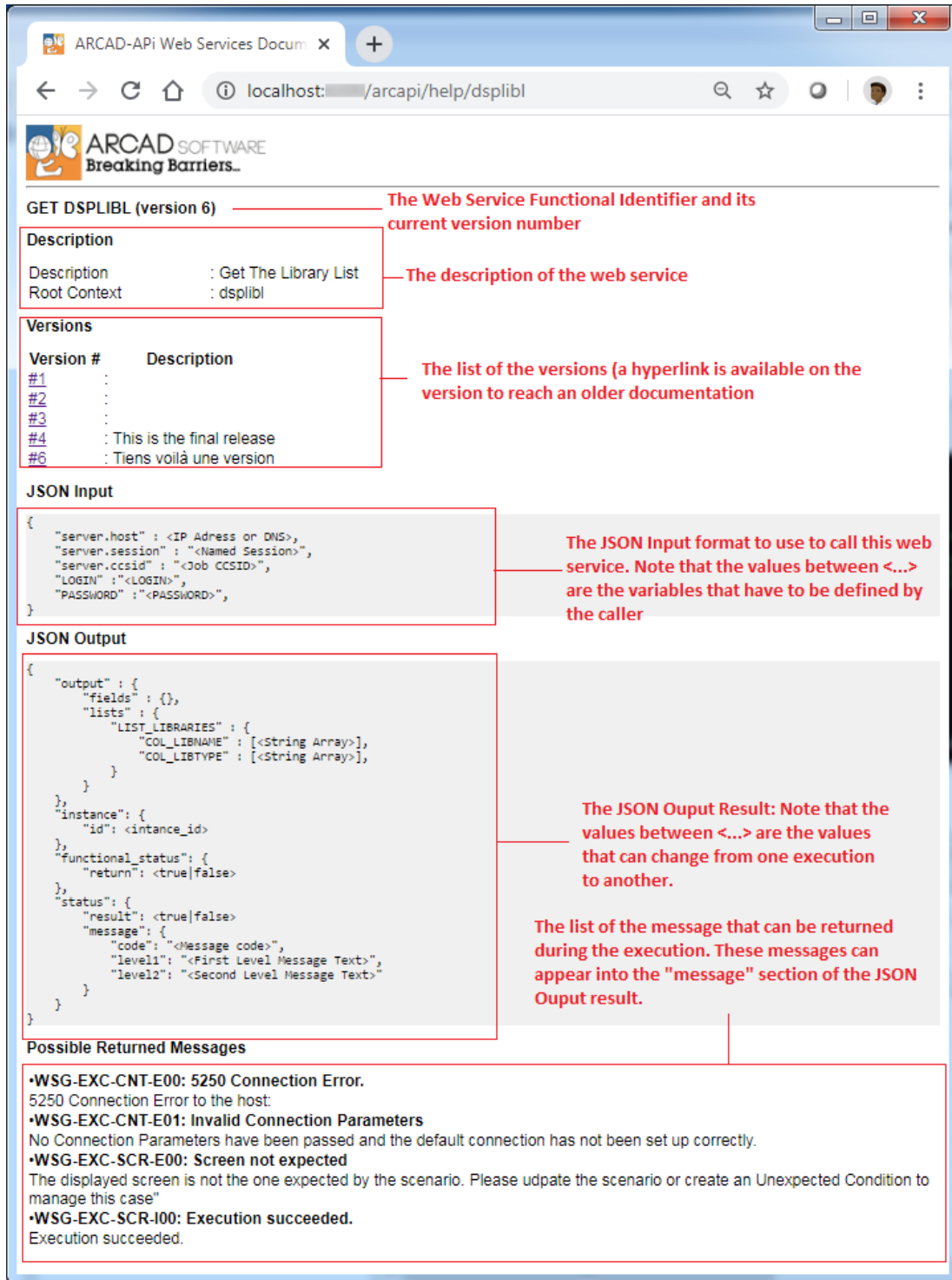
The server and port must correspond to the ARCAD-API Server on which the web services were created. The catalog is automatically generated from the services you have created and pushed to production in the studio. Click on a web service to open the full documentation.

In this page, you will see:

- the version of the web service used by the parent API,
- the description of the web service as defined in the designer,
- the root context of the web service,
- this URL that must be used,
- the JSON Input format,
- the JSON Output format,
- the list of the messages that could be returned in case of failure.

This page is directly reachable using the following URL:

<http://<arcad-api-server>:<arcad-api-port>/arcapi/help/<web-service-root-context>/<web-service-version>>



**GET DSPLIBL (version 6)** — The Web Service Functional Identifier and its current version number

**Description** — The description of the web service

Description : Get The Library List  
Root Context : dsplibl

**Versions** — The list of the versions (a hyperlink is available on the version to reach an older documentation)

Version #	Description
#1	:
#2	:
#3	:
#4	: This is the final release
#6	: Tiens voilà une version

**JSON Input** — The JSON Input format to use to call this web service. Note that the values between <...> are the variables that have to be defined by the caller

```
{
  "server.host" : <IP Address or DNS>,
  "server.session" : "<Named Session>",
  "server.ccsid" : "<Job CCSID>",
  "LOGIN" : "<LOGIN>",
  "PASSWORD" : "<PASSWORD>",
}
```

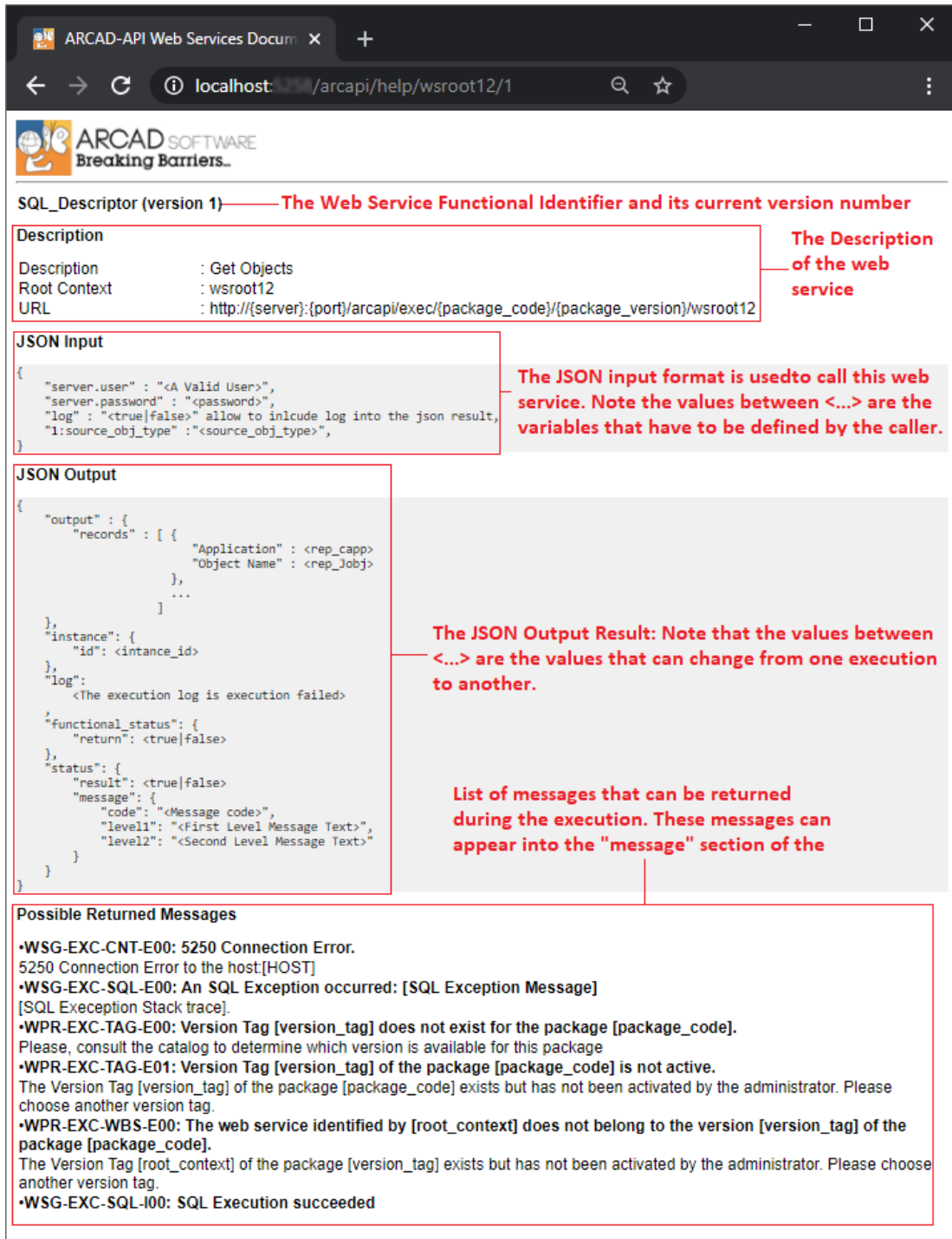
**JSON Output** — The JSON Output Result: Note that the values between <...> are the values that can change from one execution to another.

```
{
  "output" : {
    "fields" : {},
    "lists" : {
      "LIST_LIBRARIES" : {
        "COL_LIBNAME" : [<String Array>],
        "COL_LIBTYPE" : [<String Array>],
      }
    }
  },
  "instance" : {
    "id" : <instance_id>
  },
  "functional_status" : {
    "return" : <true|false>
  },
  "status" : {
    "result" : <true|false>
    "message" : {
      "code" : "<Message code>",
      "level1" : "<First Level Message Text>",
      "level2" : "<Second Level Message Text>"
    }
  }
}
```

**Possible Returned Messages** — The list of the message that can be returned during the execution. These messages can appear into the "message" section of the JSON Output result.

- WSG-EXC-CNT-E00: 5250 Connection Error.  
5250 Connection Error to the host.
- WSG-EXC-CNT-E01: Invalid Connection Parameters  
No Connection Parameters have been passed and the default connection has not been set up correctly.
- WSG-EXC-SCR-E00: Screen not expected  
The displayed screen is not the one expected by the scenario. Please update the scenario or create an Unexpected Condition to manage this case"
- WSG-EXC-SCR-I00: Execution succeeded.  
Execution succeeded.

Figure 58: Web service documentation (5250 catalog)



**ARCAD SOFTWARE**  
Breaking Barriers.

**SQL\_Descriptor (version 1)** — The Web Service Functional Identifier and its current version number

**Description**

Description : Get Objects  
 Root Context : wsroot12  
 URL : http://{server}:{port}/arcapi/exec/{package\_code}/{package\_version}/wsroot12

**The Description of the web service**

**JSON Input**

```
{
  "server.user" : "<A Valid User>",
  "server.password" : "<password>",
  "log" : "<true|false>" allow to include log into the json result,
  "1:source_obj_type" : "<source_obj_type>",
}
```

**The JSON input format is used to call this web service. Note the values between <...> are the variables that have to be defined by the caller.**

**JSON Output**

```
{
  "output" : {
    "records" : [ {
      "Application" : <rep_capp>
      "Object Name" : <rep_objj>
    },
    ...
  ],
  "instance" : {
    "id" : <intance_id>
  },
  "log" :
    <The execution log is execution failed>
  "functional_status" : {
    "return" : <true|false>
  },
  "status" : {
    "result" : <true|false>
    "message" : {
      "code" : "<Message code>",
      "level1" : "<First Level Message Text>",
      "level2" : "<Second Level Message Text>"
    }
  }
}
```

**The JSON Output Result: Note that the values between <...> are the values that can change from one execution to another.**

**List of messages that can be returned during the execution. These messages can appear into the "message" section of the**

**Possible Returned Messages**

- WSG-EXC-CNT-E00: 5250 Connection Error.  
5250 Connection Error to the host:[HOST]
- WSG-EXC-SQL-E00: An SQL Exception occurred: [SQL Exception Message]  
[SQL Exception Stack trace].
- WPR-EXC-TAG-E00: Version Tag [version\_tag] does not exist for the package [package\_code].  
Please, consult the catalog to determine which version is available for this package
- WPR-EXC-TAG-E01: Version Tag [version\_tag] of the package [package\_code] is not active.  
The Version Tag [version\_tag] of the package [package\_code] exists but has not been activated by the administrator. Please choose another version tag.
- WPR-EXC-WBS-E00: The web service identified by [root\_context] does not belong to the version [version\_tag] of the package [package\_code].  
The Version Tag [root\_context] of the package [version\_tag] exists but has not been activated by the administrator. Please choose another version tag.
- WSG-EXC-SQL-I00: SQL Execution succeeded

Figure 59: Web service documentation (SQL catalog)

# Glossary

---

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

## A

---

### application

The highest entities ARCAD can work with. On the IBM i server running ARCAD, an application corresponds to a set of libraries, referred as application operational libraries.

ARCAD identifies an application through its 3-character identifier, also called an "application code" in the user interface.

### ARCAD Repository

For an application, the repository corresponds to the list of all the components. For each component, it includes the following:

- its physical location (source, object),
- its position in the system function tree, and
- its cross-references.

This object repository allows ARCAD to know the detailed content of an application.

## B

---

## C

---

### category

See: [web service category](#)

### CCSID

Coded Character Set Identifier. Specific to IBM i. A 16-bit number that includes a specific set of encoding scheme identifiers, character set identifiers, code page identifiers, and other information that uniquely identifies the coded graphic-character representation.

### column (metadata ID)

A column ID defines the area that covers one column in a list.

## D

---

### Descriptors

Web service descriptors contain all the elements required to execute a web service on an ARCAD-API production server.

## E

---

### end of list (metadata ID)

An end of list ID defines the portion of the screen where a message appears when the end of the list is reached.

## F

---

### field (metadata ID)

A field ID defines a specific screen area that contains individual data.

## G

---

## H

---

## I

---

## J

---

## K

---

## L

---

### list (metadata ID)

A list ID defines a portion of the screen that is used to display a list of items as a table.

## M

---

### Metadata

The functional extended attribute assigned to a specific area of a screen in the 5250 emulator. Metadata allows you to assign IDs to certain fields, lists, columns or whole screens so that the area the data occupies can be identified. This ID can then be consumed by your API.

## N

---

## O

---

## P

---

### Production package

Versioned sets of promoted web services.

## Q

---

## R

---

### Registered web service

The web services available on the ARCAD-API production server.



## S

---

### Scenario

The original (or dedicated) recording of a set of actions that represents a specific activity or task that an application can carry out. The actions are required to be carried out in a given context.

### screen (metadata ID)

A screen ID defines one or more screen areas that allow you to identify a unique screen.

### screen area

A rectangle drawn on a 5250 screen that delimits the portion of text to identify as metadata.

## T

---

## U

---

### Unexpected screen action

Instructions added post-recording to a scenario that tell ARCAD-API how to react if a given result is not acquired when replaying a scenario.

## V

---

### Version tag

The ensemble of web services available for a given production package version.

## W

---

### web service category

Web service descriptors can be grouped into categories in order to manage sets of similar scenarios or scenarios with a common theme.

## X

---

## Y

---

## Z

---