

API Developer Pack

Introduction

Index

— Purpose	4
— Glossary	4
— Contents of API Developer Pack	5
• Package name.....	5
• Directory structure.....	6
— Where do I start?	7
• What is Amadeus for Developers Portal?	7
• What is SoapUI?	7
• Time to take first steps	8
— Addendum	14
• Test case description.....	14
• Adding your own WSDL file to SoapUI Project	15
• 00. Security library TestSuite	16
• Set SOAP headers groovy step	18

Document control				
Security level				
Company	Amadeus IT Group SA			
Department	Web Services Consultancy			
Author	Tomasz Radecki			
Reviewed by	Tomasz Radecki	Date	2017/07/14	
Approved by	Laurent Gagou	Date	2017/05/15	
Version	Date	Change	Comment	By
0.1	2017/05/26	Initial	Draft version	Tomasz Radecki
0.2	2017/04/25	Update	Update after review by Adam Nowik	Adam Nowik, Tomasz Radecki
0.3	2017/05/18	Update	Update of next steps section suggested by Adam	Adam Nowik, Tomasz Radecki
0.4	2017/05/26	New content	Screenshots added to section Time to take first steps.	Adam Nowik, Tomasz Radecki
0.5	2017/06/09	Update	Changing the first steps list to reflect the current state of SoapUI project	Adam Nowik, Tomasz Radecki
0.6	2017/06/23	Update	Reshaping file to be more reusable	Adam Nowik
0.7	2017/07/11	Update	Addition of 00. Security and Set SOAP Header chapters	Adam Nowik
0.8	2017/08/23	Update	Unifying document for different area's	Adam Nowik
0.9	2018/04/13	Update	Added a chapter about test case description	Adam Nowik
1.0	2019/02/19	Update	Adapting the file, changing extranet into Amadeus for developers; Fixes.	Adam Nowik, Łukasz Siemiradzki

Purpose

Purpose of this document is to introduce contents of the API Developer Pack to reader to and help him/her to take first steps with it.

Glossary

Abbreviations and terms used in the document are listed below.

IBE	Internet Booking Engine
WSDL	The Web Services Description Language is an XML-based interface definition language that is used for describing the functionality offered by a web service. The acronym is also used for any specific WSDL description of a web service (also referred to as a WSDL file), which provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns.
ZIP	ZIP is an archive file format that supports lossless data compression. A .ZIP file may contain one or more files or directories that may have been compressed.

Contents of API Developer Pack

Package name

Name of the package is created according to following pattern: `ADP_$area` where:

- `$area` indicates which area pack is dedicated to (Internet Booking Engine, Hotel, etc.)

For example, `ADP_IBE` means that the pack is dedicated to Internet Booking Engine (IBE).

Directory structure

Below you will find a snapshot of example API Developer Pack contents (the actual content of your pack may vary):

```
ADP_IBE

  \environment access
    IBE_1ASIWPOC1A_PDT.zip
    IBE_1ASIWPOC1A_WS_LIST.txt

  \guides
    Amadeus WBS Implementation Guide - IBE, MasterPricer.docx
    Amadeus WBS Implementation Guide - SOAP 4.0.pdf

  \SoapUI mock-up
    ADP_IBE.xml

API Developer Pack introduction.pdf
```

`\environment access`

This directory contains a ZIP archive with a WSDL file, for example `1ASIWPOC1A_PDT.zip`.

`\guides`

The name is self-explanatory. Here you will find:

- `Amadeus WBS Implementation Guide - IBE, MasterPricer.docx` – a document describing concept of building internet booking engine, description of shopping, booking and ticketing phases as well as particular web services which should be used at each step
- `Amadeus WBS Implementation Guide - SOAP 4.0.pdf` – a document describing how your application should establish connection and manage sessions

`\SoapUI mock-up`

This directory contains an exported SoapUI mock-up project `xml` file. It was created in order to demonstrate how internet application works from web services perspective (sequence of web service calls, argument and value feeding or passing between the calls).

Where do I start?

Start with going through basic explanation below.

What is Amadeus for Developers Portal?

Amadeus for Developers Portal (<https://developers.amadeus.com/>) is your place to go to when searching for information about Amadeus Web Services usage and related best practices at implementation time.

What is SoapUI?

According to official web page (<https://www.soapui.org/open-source.html>):

SoapUI is the world's most widely-used open source API testing tool for SOAP and REST APIs. SoapUI offers SOAP Web Services functional testing, REST API functional testing, WSDL coverage, message assertion testing and test refactoring. With over 10 years of experience backed by a vast open source community, SoapUI is the de facto method for ensuring quality when developing APIs and Web Services.

Please make sure that you are using SoapUI in Version 5.5 or above. Please also ensure that SoapUI is configured to use TLS 1.2 to use Amadeus Web Services as this is the minimal supported version of cryptographic protocols.

Time to take first steps

1. Verify that your credentials for Amadeus For Developers portal are valid by signing in and navigating around

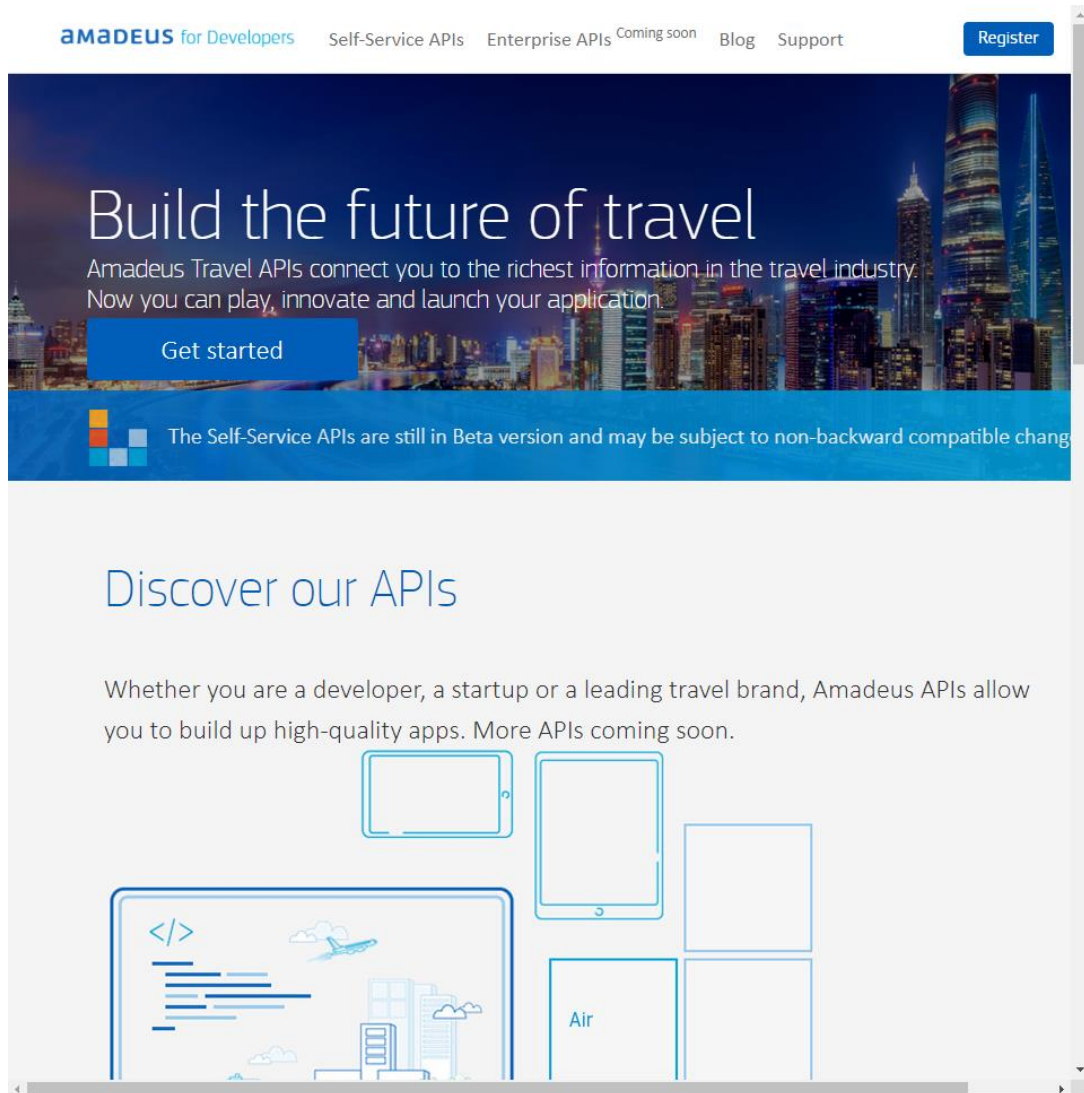


Figure 1

2. Download and install SoapUI software (for detailed instruction please refer to official web page <https://www.soapui.org/getting-started/installing-soapui.html>).

3. Run SoapUI application and then:
 - a. Import provided SoapUI mock-up project ([ADP_IBE.xml](#)).

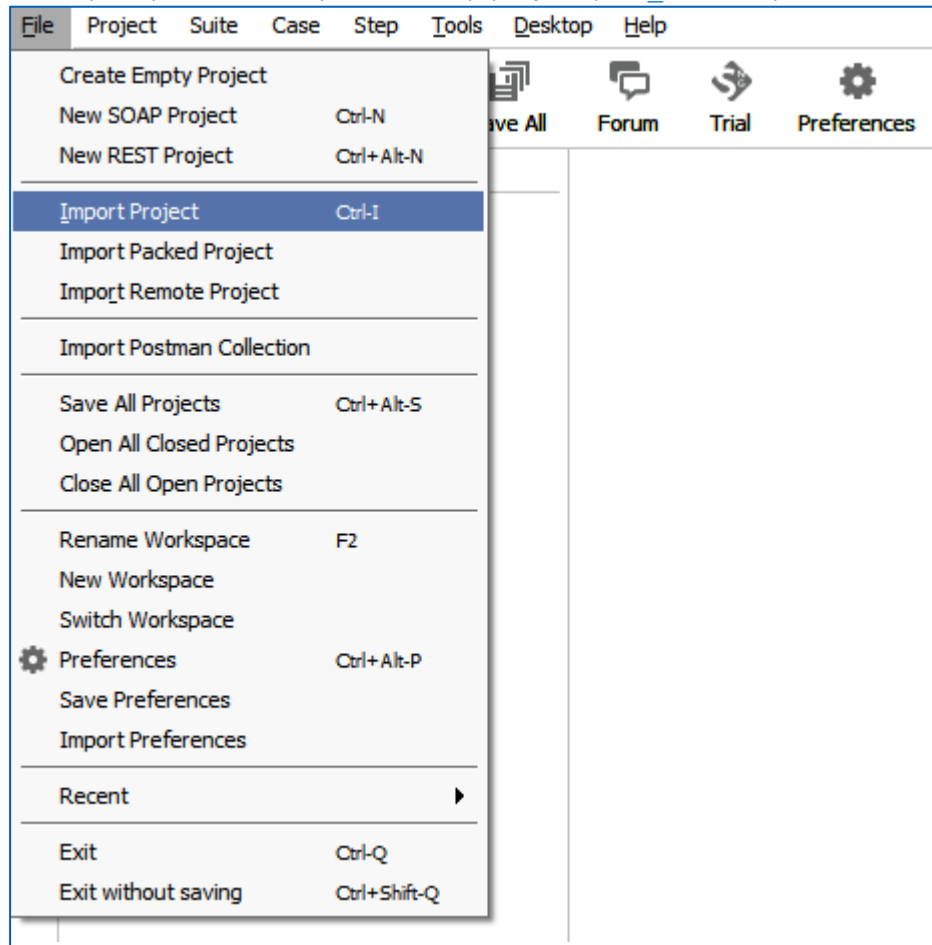


Figure 2

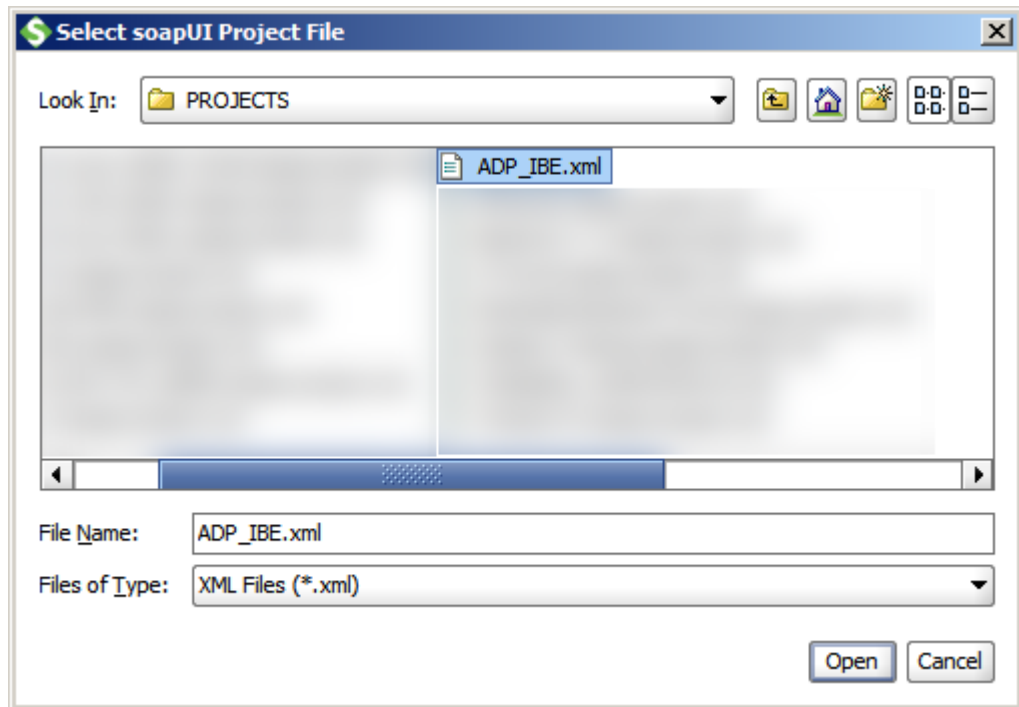


Figure 3

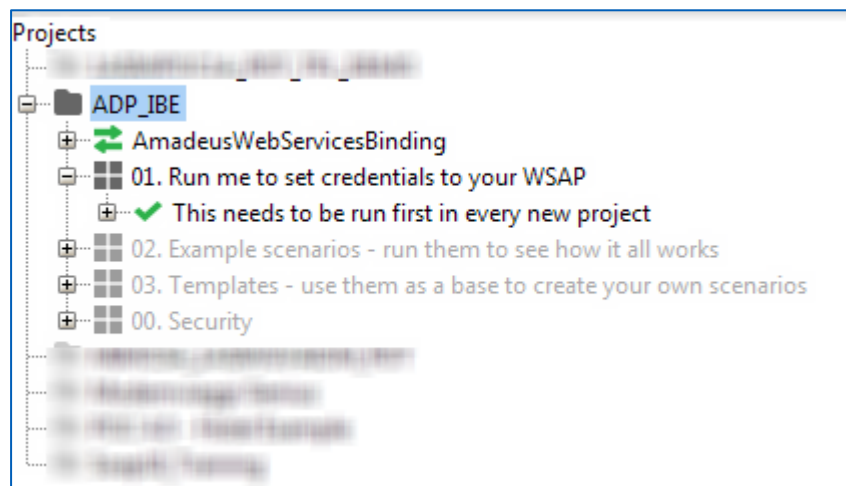


Figure 4

- b. Run the test suite named *01. Run me to set credentials to your WSAP* and fill in the popup window with test environment credentials you have received with API Developer Pack.

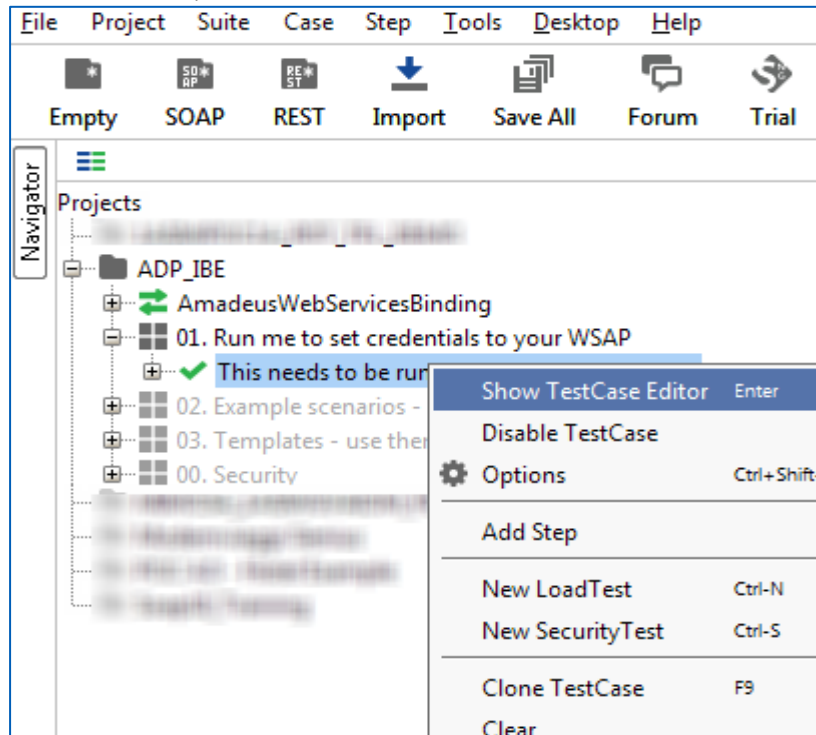


Figure 5

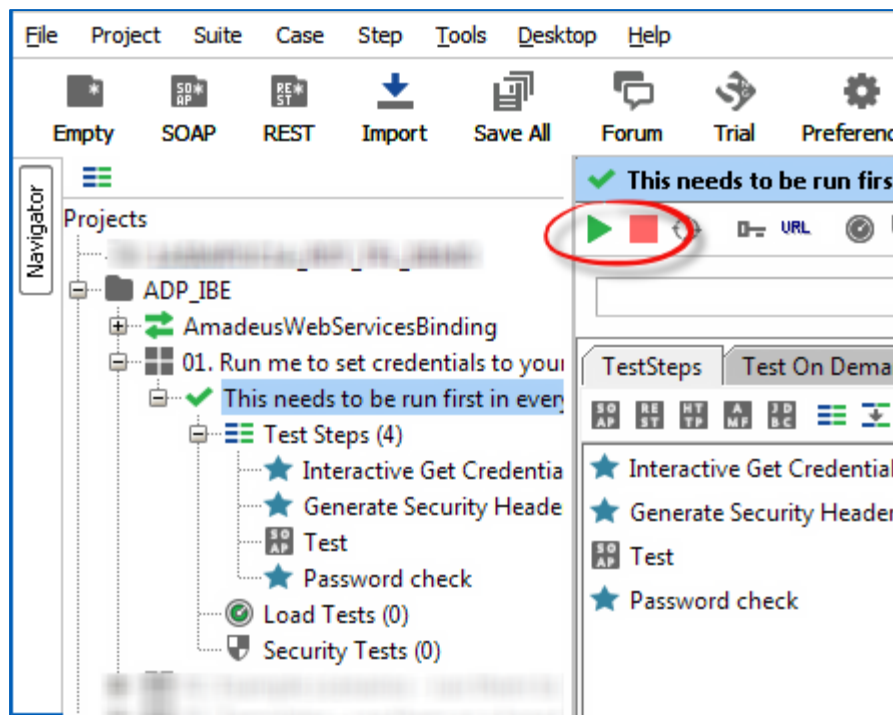


Figure 6

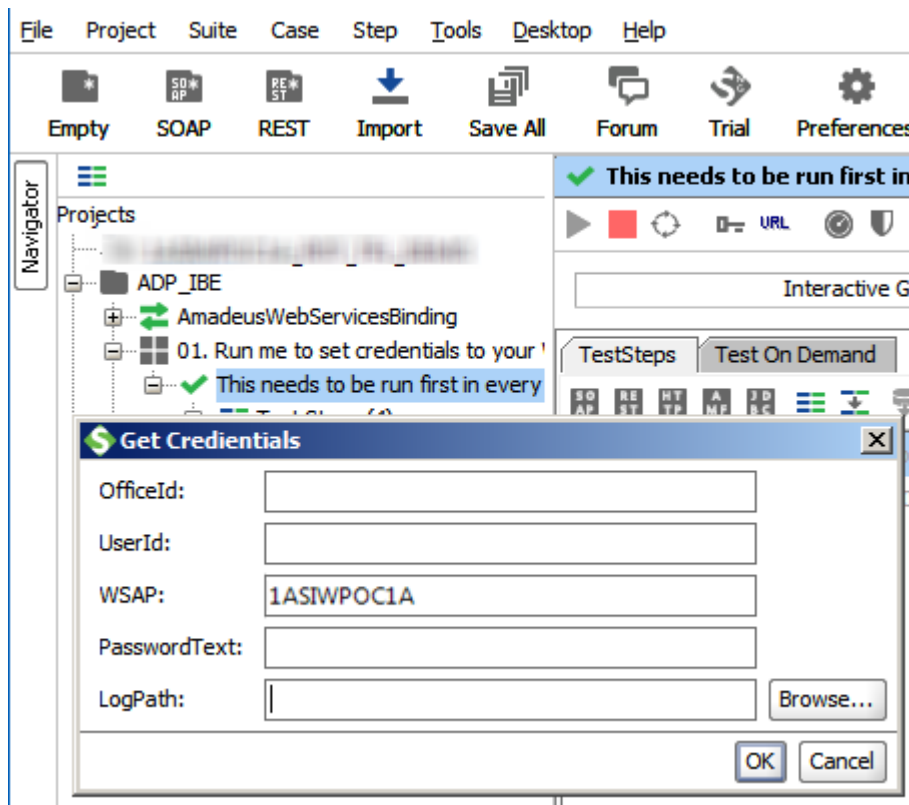


Figure 7

- c. Go to 02. Example scenarios - run them to see how it all works test suite in the SoapUI mock-up project to verify your access to test environment and see how it works.

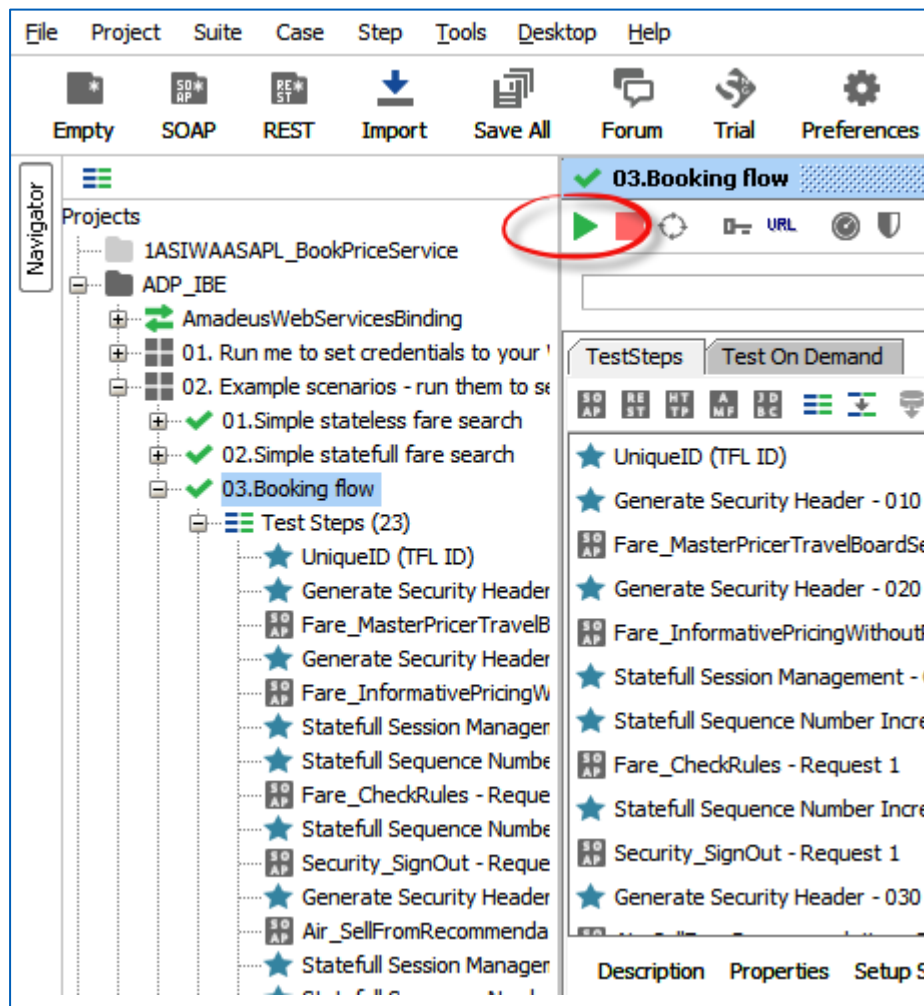


Figure 8

- d. Use *03. Templates* - use them as a base to create your own scenarios to modify, copy, and change the provided project file as you wish. Playing on your own with the Amadeus Web Services is the best way to learn.
4. Read provided documentation dedicated to building application kind your dev pack is designed to (e. g. [Amadeus WBS Implementation Guide - Internet Booking Engine with Master Pricer - v.1.2.docx](#)) and session management ([SOAP 4.0 Amadeus WBS Implementation Guide.pdf](#)) to get better understanding of the basic techniques for authentication and session handling.
5. At this point you should have general understanding how an internet booking engine works. It's time to use your favourite web service framework, WSDL and start building your own application.

Addendum

Test case description

Each Test Case has a description which tells you a bit more about it:

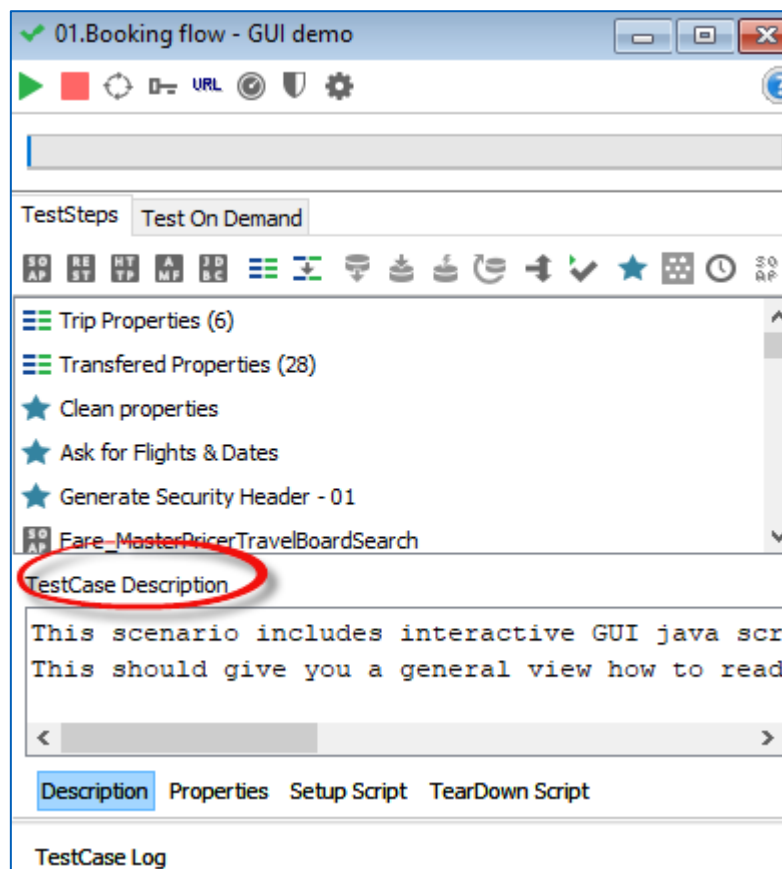


Figure 9

Adding your own WSDL file to SoapUI Project

Later on you will be equipped with your dedicated WSAP and customized WSDL file. You may then use SoapUI to mock-up web services communication with your access point. To add your WSDL file to the existing SoapUI mock-up project do the following:

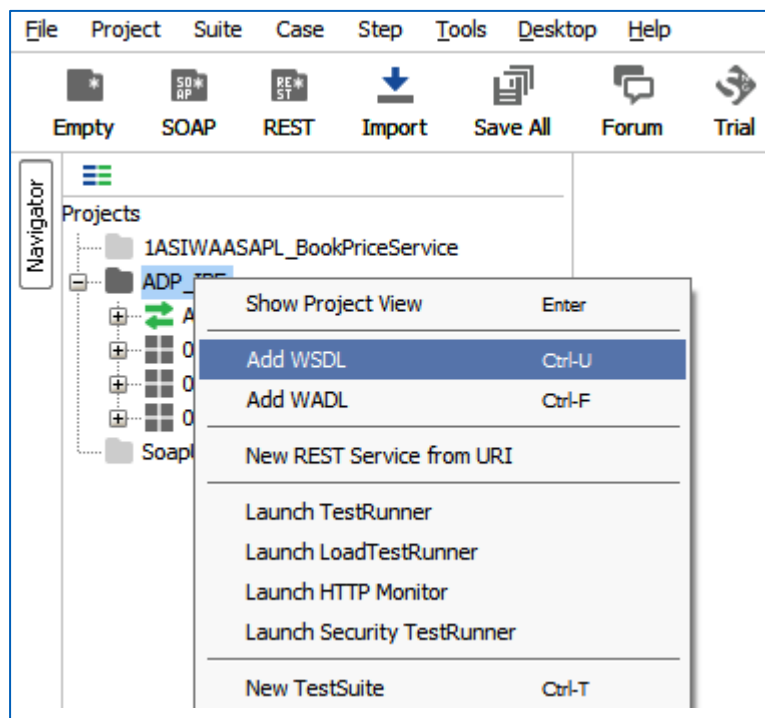


Figure 10

If a WSDL file was imported previously to the project you might be prompted to update the old WSDL definition instead of adding a new one:

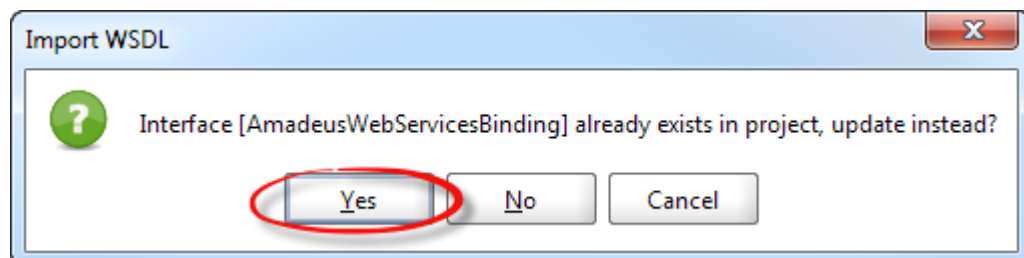


Figure 11

Choosing Yes when asked for update is the right way to go. Running the *01. Run me to set credentials to you WSAP* (Figure 7) might be useful to change the credentials for your new WSAP connection.

This way you can also add more than one WSDL file for example in situations when you use more than one interface in your project and each interface has its dedicated WSDL file.

After adding new WSDL file please make sure that the endpoint is set to Amadeus test node without the WSAP name at the end:

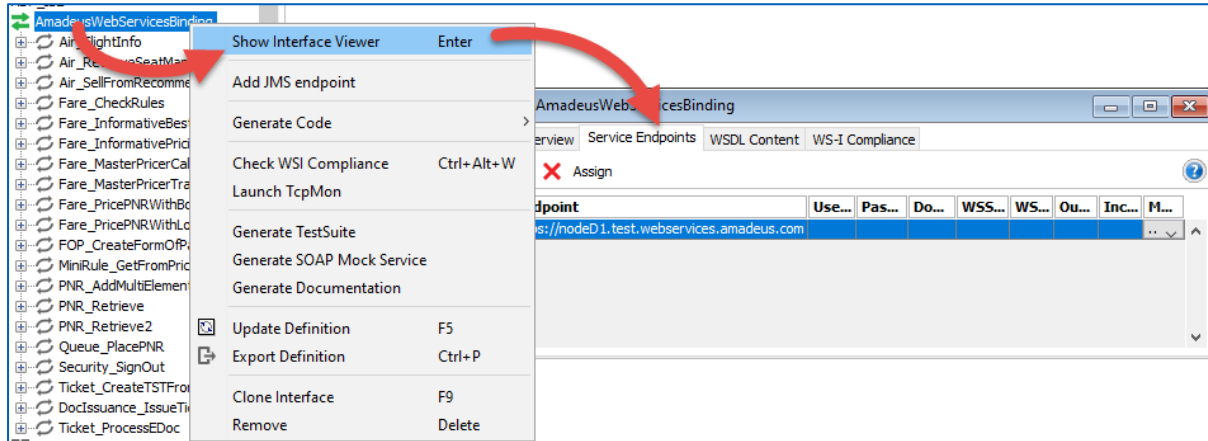


Figure 12

For example it should be:

<https://nodeD3.test.webservices.amadeus.com>

not

<https://nodeD3.test.webservices.amadeus.com/1ASIWPOC1A>

00. Security library TestSuite

SoapUI project file located in *SoapUI mock-up* subfolder has an additional TestSuite called **00. Security:**

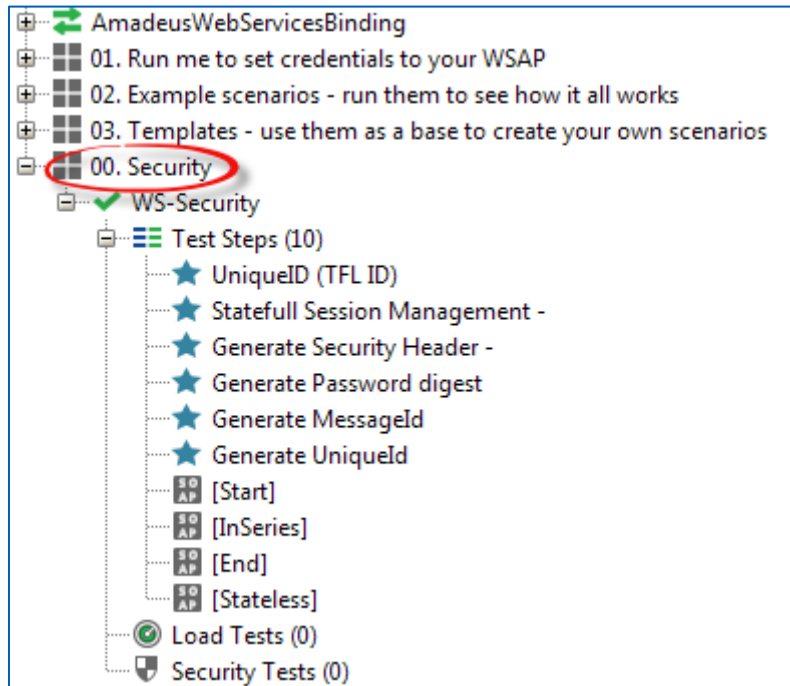


Figure 13

00. Security TestSuite is used as an external library shared by groovy steps in other TestCases in the project (*Generate Security Header*, *Set SOAP headers*) to dynamically create authorization data (*nonce*, *timestamp*, *sha-1 password digest*, etc) and update contents of the SOAP requests with correct SOAP Headers.

Modification of this TestSuite name or its content may cause malfunctioning of the groovy scripts which then may cause errors that will not be supported by Amadeus.

If you plan to use SOAP header update scripts you should copy *00. Security* TestSuite to your project.

Set SOAP headers groovy step

To make things easier we have created a groovy script which automates and speeds-up the creation of SOAP 4.0 protocol session management specific to Amadeus Web Services.

The *Set SOAP headers* groovy step is located in 03. Templates -> 00. Misc:

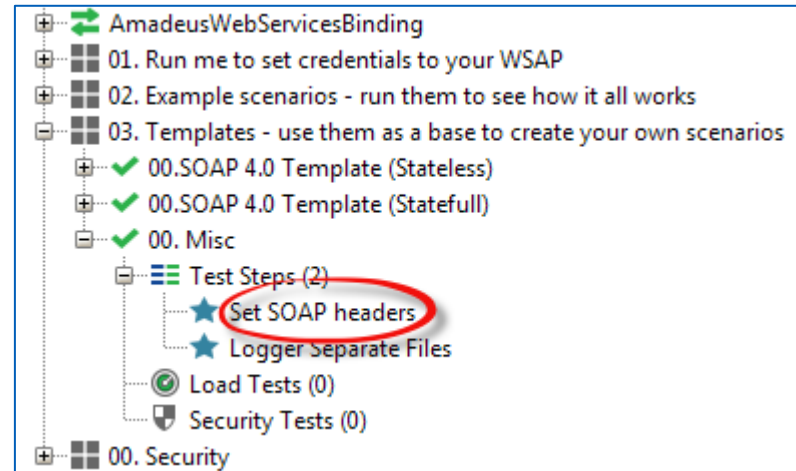


Figure 14

Script manual:

1. Create a TestSuite with the desired scenario (TestCase) of SOAP requests. Set SOAP requests in right order for your scenario.

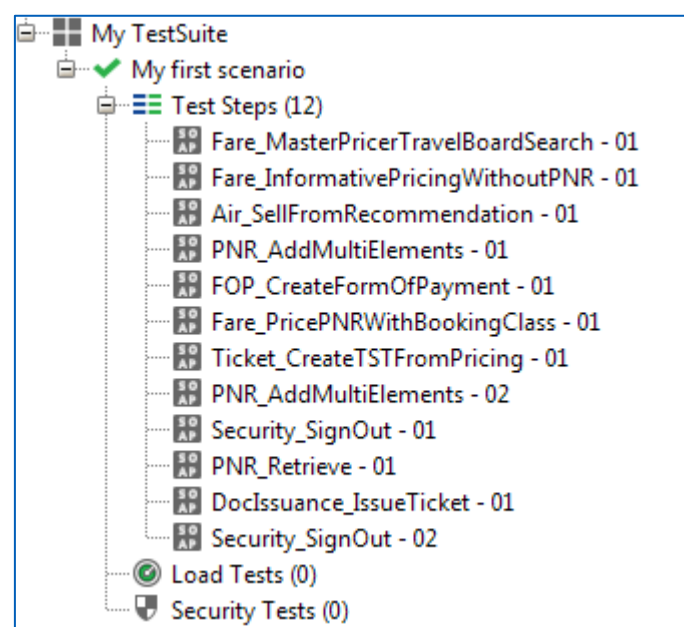


Figure 15

2. Copy the *Set SOAP headers* groovy step to your scenario. Decide which requests should be *stateless* and which should *open*, *close* or be *InSeries* in the session:
 - a. In the *session opening* request name add *[Start]* string.
 - b. In the *session closing* request name add *[End]* string.
 - c. Requests located between *session opening* (point a.) and *session closing* (point b.) will be automatically set as *InSeries*. No need to change requests' name here.
 - d. Requests NOT located between *session opening* (point a.) and *session closing* (point b.) will be automatically set as *stateless*. Again there is no need to change requests' name.

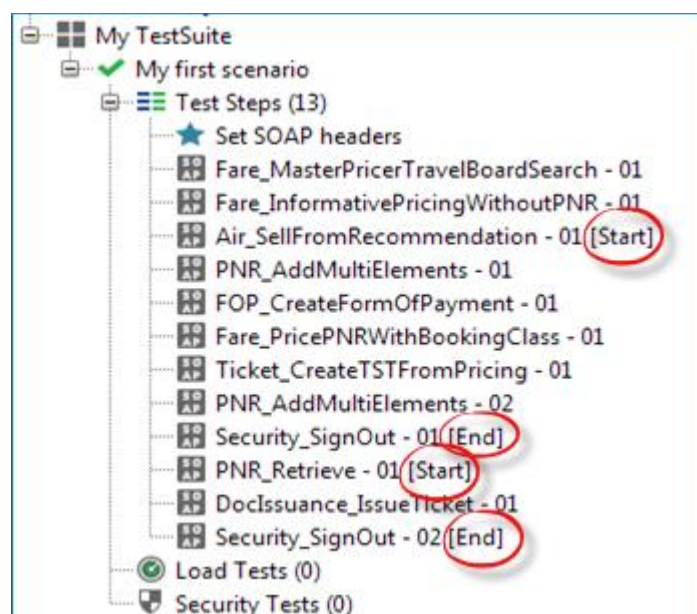


Figure 16

amadeus

Important: Full description of Amadeus session management and differences between stateless and stateful requests can be found in [Amadeus WBS Implementation Guide - SOAP 4.0.pdf](#) included in this DEV PACK. Please, read it carefully.

3. Run the *Set SOAP headers* groovy step. You should see that new groovy steps were automatically added to your scenario:
 - a. *Generate Security Header* – responsible for authorization data generation such as: *Nonce*, *MessageId*, *Created (time stamp)*, *SHA1 Password Digest*. This groovy step should be present before each *stateless* and *session opening ([Start])* request.
 - b. *Statefull Session Management* – responsible for capturing the *SecurityToken* and *SessionId* located in the response to *session opening* request. Due to the fact that it interprets the response it needs to be located right after every *session opening* request.

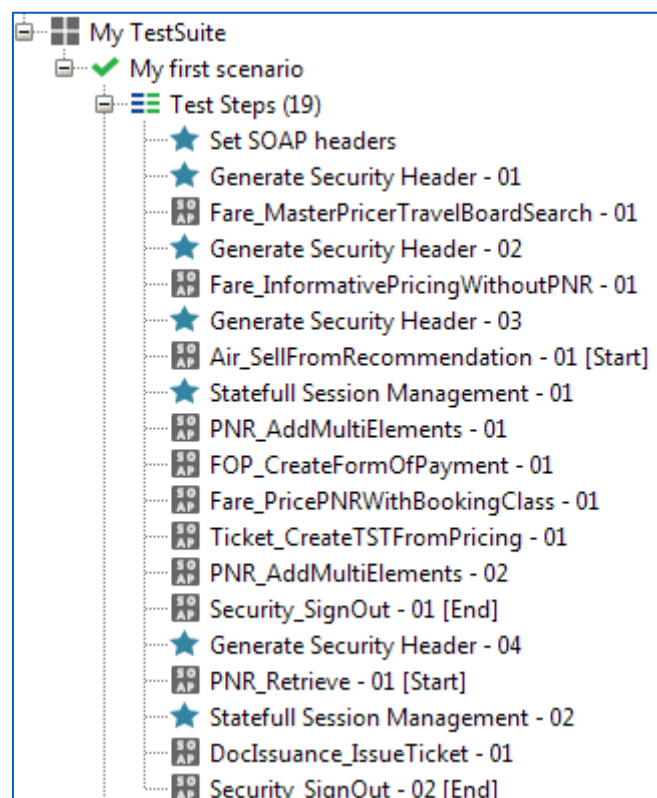


Figure 17

4. *Set SOAP headers* should also update the SOAP Header part of all your SOAP requests. Headers are different depending on a fact whether they are located in *stateless*, *session opening/closing* or *InSeries* requests.
5. Next steps are up to you – fill the requests with bodies of the messages you want to send. You can check example scenarios for inspiration, we recommend reading the implementation guides dedicated to particular web services, where examples of requests can be found. Implementation guides for specific web services are located in the Extranet portal.

Important: Remember that after every change in scenario structure (addition, deletion or changing the order of SOAP requests) the *Set SOAP headers* step needs to be run again to adjust the session management in your scenario.

Important: Remember that if you use the *Set SOAP headers* groovy step the *00. Security* TestSuite needs to be present in your SoapUI project.