MYOB Advanced

Part 1 Data Retrieval: Basic

Last Updated: 31 October 2019

Copyright 2019 MYOB Technology Pty Ltd

Contents

Copyright 2019 MYOB Technology Pty Ltd

Copyright 2019 MYOB Technology Pty Ltd

Introduction

External systems can use MYOB Advanced ERP integration interfaces to access the business functionality and data of MYOB Advanced ERP. This course introduces the following integration interfaces:

- The Open Data (OData) interface
- The contract-based REST API
- The contract-based SOAP API
- The screen-based SOAP API

This course is intended for developers who need to create applications that interact with MYOB Advanced ERP.

The course is based on a set of examples of web integration scenarios that demonstrate the processes of developing a client application that uses MYOB Advanced ERP integration interfaces. The course gives you ideas about how to develop your own applications by using the OData interface and the REST and SOAP APIs. It demonstrates the main elements of the OData interface and the web services APIs and their use in typical tasks that integrate MYOB Advanced ERP with third-party applications. As you go through the course, you can complete the examples for a particular integration interface or for multiple integration interfaces.

After you complete all the lessons of the course, you will be familiar with the basic techniques of data retrieval through the MYOB Advanced ERP OData interface and web services APIs.

Copyright 2019 MYOB Technology Pty Ltd

How to Use This Course

To complete the course, you will complete the lessons from each part of the course in the order in which they are presented and pass the assessment test. More specifically, you will do the following:

1. Complete *Course Prerequisites*, and carefully read *Initial Steps*.
2. Complete the lessons in all parts of the training guide. In each lesson, you should review the description of the lesson, at least one of the examples for the integration interface that you are interested in, and the lesson summary. You can skip the other examples of the lesson, or you may prefer to review multiple examples in the lesson if you are looking for the best solution for your integration scenario.

After you pass the certification test, you will be given the Acumatica University certificate of course completion.

What Is in a Part?

The first part of the course explains how to configure a third-party application to use the MYOB Advanced ERP integration interfaces.

The second and third parts of the course are dedicated to the implementation of particular web integration scenarios that you may need to implement in a third-party application that integrates an external system with MYOB Advanced ERP.

The fourth part of the course describes how to include in a customization project the items that you have created in MYOB Advanced ERP for integration with an external system.

Each part of the course consists of lessons you should complete.

What Is in a Lesson?

Each lesson is dedicated to a particular web integration scenario that you can implement by using the OData interface and web services APIs. Each lesson consists of a brief description of the web integration scenario and examples of the implementation of this scenario.

The lesson may also include *Additional Information* topics, which are outside of the scope of this course but may be useful to some readers.

Each lesson ends with a *Lesson Summary* topic, which summarizes the possible options for the implementation of the web integration scenario with different integration interfaces.

What Are the Documentation Resources?

All the links listed in the *Related Links* sections refer to the documentation available on the *https://help.acumatica.com/* website. These and other topics are also included in the MYOB Advanced ERP instance, and you can find them under the **Help** menu.

Copyright 2019 MYOB Technology Pty Ltd

Course Prerequisites

To complete this course, you should be familiar with the basic principles of MYOB Advanced ERP and of the creation of generic inquiries in MYOB Advanced ERP.

You should complete this course on MYOB Advanced ERP 2019. Before you start with this course:

1. You need to deploy an instance of MYOB Advanced ERP with the name *MyStoreInstance.* For information on how to deploy this instance, see *Deploying an MYOB Advanced ERP Instance for the Training Course*.
2. The Postman application should be installed on your computer if you are going to complete the examples that illustrate the use of the OData protocol and the REST API. To download and install Postman, follow the instructions on *https://www.getpostman.com/apps*.
3. Microsoft Visual Studio 2015 or later should be installed on your computer if you are going to complete the examples that illustrate the use of the SOAP API.

   The instructions in this guide are designed for Microsoft Visual Studio 2017. If you use a different version of Visual Studio, the menu paths and the user interface may differ.

4. You must have HTTP access from the computer where you work with the examples to the MYOB Advanced ERP instance so you can work with the integration services.

Copyright 2019 MYOB Technology Pty Ltd

Deploying an MYOB Advanced ERP Instance for the Training Course

You deploy an MYOB Advanced ERP instance and configure it as follows:

1. Open the MYOB Advanced ERP Configuration Wizard, and deploy a new application instance as follows:
    a. On the **Database Configuration** page of the MYOB Advanced ERP Configuration Wizard, type the name of the database: `MyStoreInstance`.
    b. On the **Tenant Setup** page, set up one tenant with the *I100* data inserted by specifying the following settings:
        - **Login Tenant Name**: `MyStore`
        - **New**: Selected
        - **Parent Tenant ID**: 2
        - **Visible**: Selected

    The system creates a new MYOB Advanced ERP instance, adds a new tenant, and loads the selected data.
2. Sign in to the new tenant by using the following credentials:
    - Login: `admin`
    - Password: `setup`

    Change the password when the system prompts you to do so.
3. Click the user name in the top right corner of the MYOB Advanced ERP window, and click **My Profile**. On the **General Info** tab of the User Profile (SM203010) form, which opens, select *MYSTORE* in the **Default Branch** box; then click **Save** on the form toolbar. In subsequent sign-ins to this account, you will be signed in to this branch.

Copyright 2019 MYOB Technology Pty Ltd

Initial Steps

This part of the course provides an overview of MYOB Advanced ERP integration interfaces (the OData interface, the contract-based REST and SOAP APIs, and the screen-based SOAP API) and a description of the sample integration solution, MyBIIntegration, that you will create when completing this course.

After you finish this part of the course, you will be ready to start developing the MyBIIntegration solution, which will be integrated with MYOB Advanced ERP. The examples of this training course show step- by-step development of this solution with different integration interfaces.

Copyright 2019 MYOB Technology Pty Ltd

Overview of the MYOB Advanced ERP Integration Services

MYOB Advanced ERP provides integration with external data sources and third-party systems through integration services. These integration services include the import and export of data by means of integration scenarios and interaction with external systems through the OData protocol, web services APIs, and the mobile API. External applications and systems that use the MYOB Advanced ERP integration services can access the data managed by MYOB Advanced ERP and the business functionality of MYOB Advanced ERP. For example, you can integrate MYOB Advanced ERP with a business intelligence (BI) system, an eCommerce system, or an online store system.

In the diagram below, the integration services that are covered in this training course are highlighted with blue.

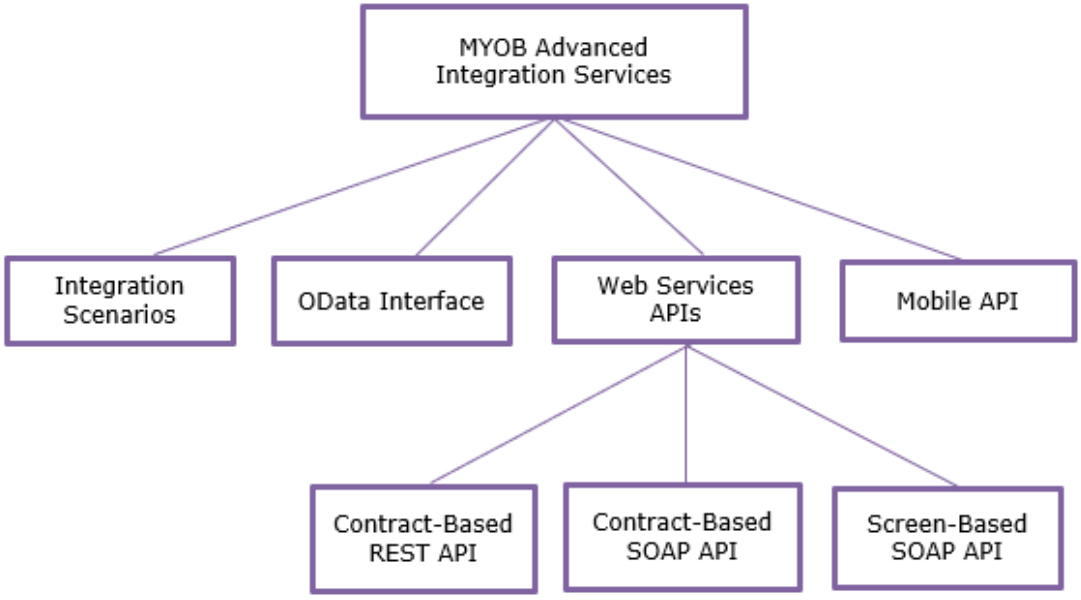


Figure: MYOB Advanced ERP integration services

This training course covers the following integration interfaces:

- OData interface, which operates through the OData protocol with the generic inquiries that are configured in MYOB Advanced ERP.
- Contract-based REST API, which operates through the REST service with business logic objects and their properties and methods, which do not depend on MYOB Advanced ERP forms. (In this context, *contract-based* means based on the object model the web service API provides.)
- Contract-based SOAP API, which operates with the same business logic objects and their properties and methods as the contract-based REST API does but uses the SOAP interface.
- Screen-based SOAP API, which works with MYOB Advanced ERP forms by using the fields and actions available on these forms.

Copyright 2019 MYOB Technology Pty Ltd

Through the OData protocol and the web services of MYOB Advanced ERP, external systems can retrieve data records from MYOB Advanced ERP. With the web services, external systems can also process these records and save the new or updated records to MYOB Advanced ERP.

Copyright 2019 MYOB Technology Pty Ltd

OData Interface

In MYOB Advanced ERP, by using the Open Data Protocol (OData), you can expose to external systems the results of generic inquiries. MYOB Advanced ERP supports OData Version 3.0.

Any generic inquiry that has been created on the Generic Inquiry (SM20800) form can be exposed via OData. To expose the generic inquiry through OData, you select the **Expose via OData** check box for the inquiry on the Generic Inquiry form.

You can use the OData interface to retrieve the data and analyze it in an external application, such as a business intelligence (BI) application. You cannot edit the data in MYOB Advanced ERP through the OData interface.

Related Links

OData Version 3.0 Specification
OData Support

Copyright 2019 MYOB Technology Pty Ltd

Contract-Based REST and SOAP APIs

The contract-based web services APIs operate with business logic objects that do not depend on MYOB Advanced ERP forms or their properties and methods. (In this context, *contract-based* means based on the object model the web services API provides.) Each contract of the web service is fixed and does not change based on system customization, localization, or any other changes made to MYOB Advanced ERP.

You can work with the contract-based web services through either the REST interface or the SOAP interface.

You access the contract-based SOAP or REST API through endpoints, which you can configure on the Web Service Endpoints (SM207060) form. An *endpoint* is an entry point to the MYOB Advanced ERP web services. For each endpoint that a web service API provides, a *contract* of the endpoint defines the entities, along with their actions and fields, that are available through the endpoint and the methods that you can use to work with these entities. For the information about the API entities, fields, and actions, see *API Entities, Fields, and Actions* in the documentation.

You can use two types of endpoints to access the web services: system and custom. The system endpoints are preconfigured in the system and have the *Default* name. Each of these endpoints has a predefined contract, which includes the API that is preconfigured in the system. You cannot change the contract of a system endpoint. By default, there are no custom endpoints in the system. If the API provided by the system endpoint is not sufficient for the requirements of your application, you can create a custom endpoint.

Related Links

Contract-Based Web Services API
Endpoints and Contracts
Comparison of Contract Versions
Comparison of System Endpoints
API Entities, Fields, and Actions

Copyright 2019 MYOB Technology Pty Ltd

Screen-Based SOAP API

The screen-based SOAP API works with MYOB Advanced ERP forms. That is, it provides API objects and methods for working with elements on MYOB Advanced ERP forms.

To upload data to and retrieve data from MYOB Advanced ERP by using the screen-based SOAP API, you should define the sequence of commands for the system for working with elements on an MYOB Advanced ERP form. When you enter a data record into the system manually, you perform a sequence of actions. You open the needed data entry form, and as you add a new record, you use the UI elements one by one—that is, you type text, select values from combo boxes, clear or select check boxes, and click buttons. In the sequence of commands for the web services, you compose the same sequence of actions, specifying a command for each user action on the form.

This sequence of commands is similar to the sequence of commands you configure when creating import and export scenarios.

The WSDL description of the MYOB Advanced ERP screen-based web services API contains the descriptions of the API objects and methods that you can use to access MYOB Advanced ERP forms. Because of the connection of the API with MYOB Advanced ERP forms, each generated WSDL description of this API reflects the current state of the system. That is, the WSDL description does not include any changes made to the system after the WSDL file was generated, and each time you change the system, you should regenerate the WSDL description and update your application accordingly.

You can prevent breaking changes in your application and omit regeneration of the WSDL description for each change in the system by using the screen-based API wrapper. For details about the wrapper, see *Screen-Based API Wrapper*.

For the information about the API objects, see *API Objects Related to MYOB Advanced ERP Forms* in the documentation.

Related Links

Screen-Based Web Services API
API Objects Related to MYOB Advanced ERP
forms Screen-Based API Wrapper

Copyright 2019 MYOB Technology Pty Ltd

Company Story and MyBIIntegration Application

In this course, you will simulate the integration of MYOB Advanced ERP with a business intelligence (BI) application that is used by a small retail company, MyStore. This company is a single business entity that has no branches or subsidiaries. MyStore uses MYOB Advanced ERP for customer management and inventory management.

MyStore plans to extend its business and needs analytic data about its customers and the goods in the store. MyStore needs to investigate the options available in MYOB Advanced ERP for integration with BI applications. The integration application, which MyStore is developing, should retrieve information about customers and stock items from MYOB Advanced ERP.

The MyBIIntegration application, which you will build as you complete the course, will integrate MYOB Advanced ERP with the BI application of the MyStore company. For the implementation of the MyBIIntegration application, the MyStore company can use one of the following interfaces:

- OData interface
- Contract-based REST API
- Contract-based SOAP API
- Screen-based SOAP API

The examples of this course show the implementation of the MyBIIntegration application with all of these interfaces. The following diagram shows how the MyBIIntegration integration application fits in the integration of the MyStore BI application with MYOB Advanced ERP.

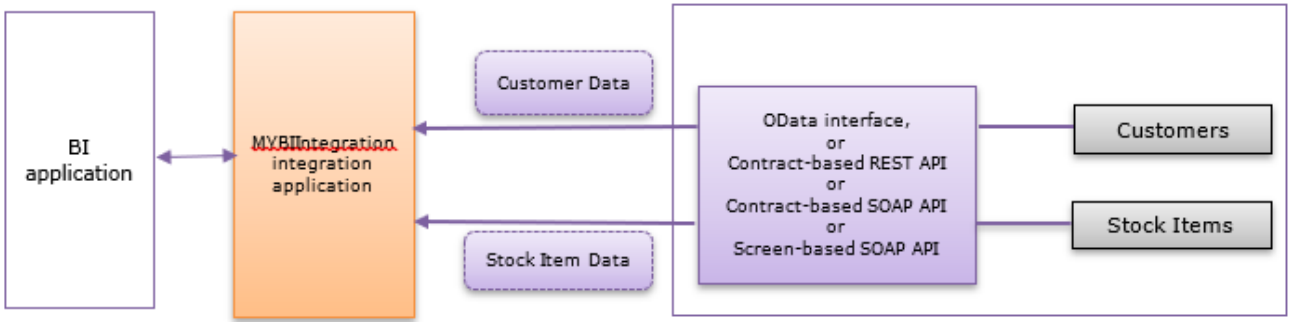


Figure: Integration of the MyStore BI application and MYOB Advanced ERP

Integration Requirements

The BI application of the MyStore company will be used by the marketing managers and warehouse managers to analyze existing customers and review the item availability in warehouses.

The MyBIIntegration application should implement integration with MYOB Advanced ERP to support the following usage scenarios in the BI application:

- Retrieval of the list of customers with contact and address details of each customer
- Retrieval of the information about availability of stock items in warehouses

In this course, you do not implement the BI application itself; instead, you implement the integration part between the BI application and MYOB Advanced ERP, which provides the support for the listed scenarios in the BI application. (The implementation of the BI application is outside of the scope of this course.)

Copyright 2019 MYOB Technology Pty Ltd

Part 1: Client Application Configuration

In this part of the guide, you will learn how to configure a client application to use the MYOB Advanced ERP integration interfaces. You will also learn how to sign in to and sign out from MYOB Advanced ERP in the client application.

By completing the lessons of this part, you will configure the MyBIIntegration application, which uses the integration interface that you select, to sign in to and sign out from MYOB Advanced ERP.

Copyright 2019 MYOB Technology Pty Ltd

Lesson 1.1: Configuring an Integration Application

Before you can retrieve data from MYOB Advanced ERP by using the integration interfaces provided by MYOB Advanced ERP, you may need to configure MYOB Advanced ERP and the integration application for use of this interface. You will perform this configuration for different integration interfaces, as described in the following sections.

Configuration for the Use of the OData Interface

For the purposes of this training course, the integration with MYOB Advanced ERP through OData does not require any prior configuration of MYOB Advanced ERP or the integration application. To allow access to the OData interface for your real integration application, you may need to configure cross-origin resource sharing (CORS). For a brief description of CORS support, see *Additional Information: CORS Support for OData*.

Configuration for the Use of the REST API

If you are integrating MYOB Advanced ERP with an external system through the contract-based REST API, you do not need to specifically configure MYOB Advanced ERP and the integration application if the application connects to MYOB Advanced ERP through a system endpoint.

If you need to use a custom endpoint, you need to create it in MYOB Advanced ERP, which is outside of the scope of this course. For short information about custom endpoints, see *Additional Information: Custom Endpoints and Endpoint Extensions*.

Configuration for the Use of the Contract-Based SOAP API

If you are integrating MYOB Advanced ERP with an external system through the contract-based SOAP API, you need to configure the Visual Studio project of the integration application so that you can use the objects and methods provided by the contract-based SOAP API in the application. In this lesson, you will configure the MyBIIntegration application to use the API of the system endpoint.

Also, you can configure a custom endpoint in MYOB Advanced ERP and use it in the integration application, which is outside of the scope of this course. For a brief description of custom endpoints, see *Additional Information: Custom Endpoints and Endpoint Extensions* in this guide.

Configuration for the Use of the Screen-Based SOAP API

For the integration of MYOB Advanced ERP with an external system through the screen-based SOAP API, you need to configure the Visual Studio project of the integration application so that you can use the objects and methods provided by the screen-based SOAP API in the application.

In MYOB Advanced ERP, you can also configure the web service API that should be included in the WSDL description of the web service, which is outside of the scope of this course. For details about the web service configuration in MYOB Advanced ERP, see *Additional Information: WSDL File of the Screen-Based Web Service*.

Lesson Objectives

In this lesson, you will learn how to:

- Import WSDL descriptions of the contract-based SOAP and screen-based SOAP web services into a Visual Studio project.
- Configure the integration application to use cookies for the sign-in to the contract-based SOAP service.

Copyright 2019 MYOB Technology Pty Ltd

Example 1.1.1. Configuring the Client Application (Contract-Based SOAP)

In this example, you will import the WSDL description of the *Default/18.200.001* system endpoint into a Visual Studio project.

You will use the Web Service Endpoints (SM207060) form to obtain the URL of the web service endpoint.

Importing the WSDL File into a Visual Studio Project

Proceed as follows:

Instead of creating a Visual Studio project, you can use the solution provided with this course (`MyBIIntegration\CBAPI\MyBIIntegration.sln`). This solution is configured for this course and already contains all the methods that are used in this course. You can use this solution for testing.

1. Open Visual Studio, and create a new Visual C# console application with the *MyBIIntegration* name.

To create a console application, click **File** > **New** > **Project** on the main menu of Visual Studio. In the **New Project** dialog box, which appears, select the *Console App (.Net Framework)* template in the Visual C# installed templates, specify the name and location of the application, and click **OK**.

The MyBIIntegration application that you have created contains the `Program.cs` file with the `Program` class.

2. Add to the project a reference to the MYOB Advanced ERP web service as follows:

a. On the main menu of Visual Studio, click **Project** > **Add Service Reference**.

b. In the **Address** box of the **Add Service Reference** dialog box, which opens, specify the URL of the *Default* endpoint of the *18.200.001* version (see Item 1 in the following screenshot).

To copy the URL of the service, on the Web Service Endpoints (SM207060) form, select *Default* in the **Endpoint Name** box and *18.200.001* in the **Endpoint Version** box, click **View Endpoint Service** > **WSDL** on the form toolbar, and copy the URL from the address line in the browser for the page that opens. In this example, the URL is *http://localhost/MyStoreInstance/entity/Default/18.200.001?wsdl*.

c. Click **Go** (Item 2) to make Visual Studio connect to the web service.

d. In the **Namespace** box, type `Default` (3). This name will be used as a namespace for the web service classes generated by Visual Studio based on the WSDL description of the service.

e. Click **OK** (4) to close the dialog box and add to the project the reference to the specified service.

Copyright 2019 MYOB Technology Pty Ltd

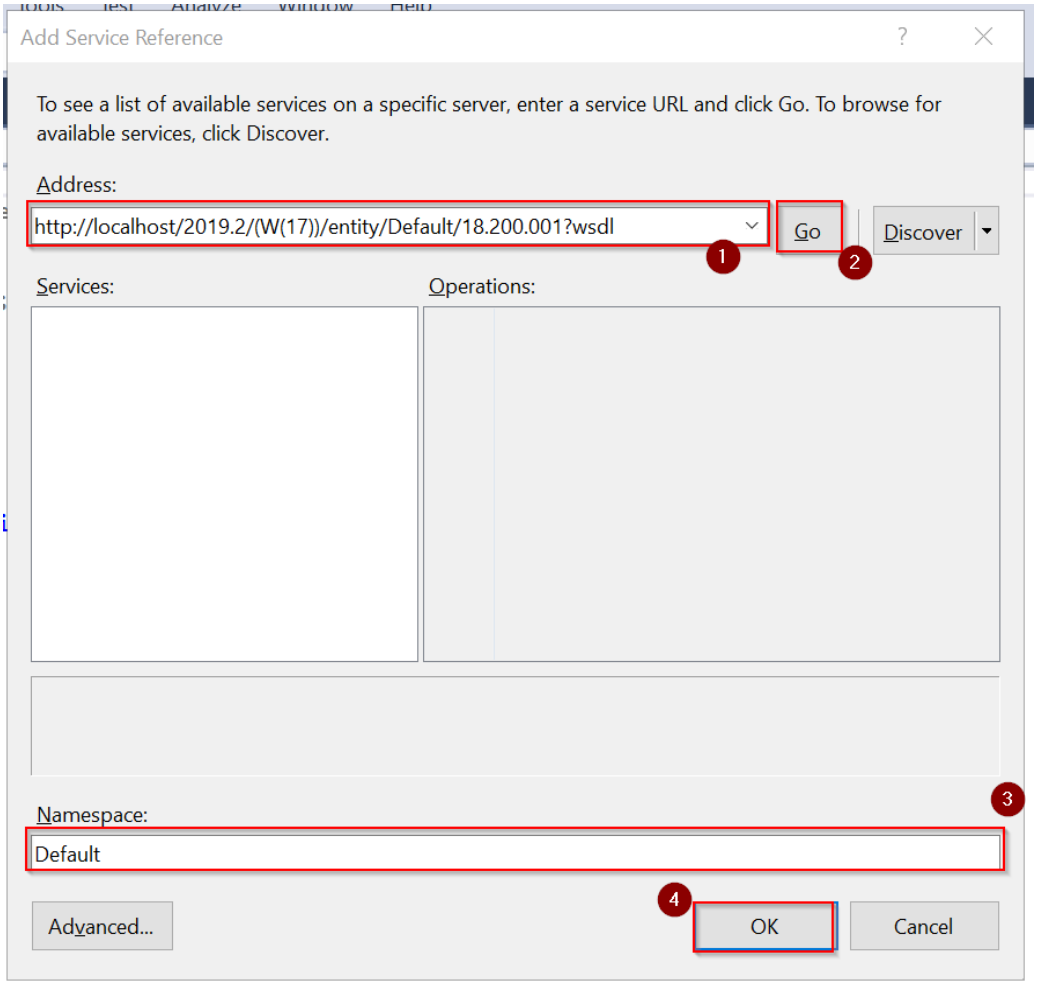


Figure: Add Service Reference dialog box

Visual Studio adds to the project the *Default* service reference in the `Service References` project folder, as shown in the following screenshot. Double-click the *Default* service reference to open the Object Browser and view the list of objects and methods available through the service.

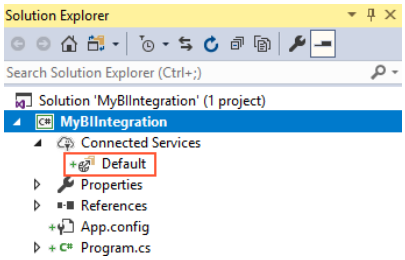


Figure: Solution Explorer

3. Modify the `app.config` file of the project as follows. Cookies are required for the client application to sign in to MYOB Advanced ERP.

Copyright 2019 MYOB Technology Pty Ltd

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>...</startup>
    <system.serviceModel>
        <bindings>
            <basicHttpBinding>
                <binding name="DefaultSoap" allowCookies ="true"/>
            </basicHttpBinding>
        </bindings>
        <client>
            <endpoint address="http://localhost/2019.2/entity/Default/18.200.001"
                binding="basicHttpBinding" bindingConfiguration="DefaultSoap"
                contract="Default.DefaultSoap" name="DefaultSoap" />
        </client>
    </system.serviceModel>
</configuration>
```

To make API calls to MYOB Advanced ERP through HTTPS, you can use the following configuration in the `app.config` file. (Here you use the HTTPS address of the endpoint instead of the HTTP address, and use the `Transport` security mode, which indicates that API calls to MYOB Advanced ERP are made through HTTPS.)

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  ...
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="DefaultSoap" allowCookies="true"
         maxReceivedMessageSize="6553600">
          <security mode="Transport" />
        </binding>
        ...
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address=
       "https://localhost/MyStoreInstance/entity/Default/18.200.001"
       binding="basicHttpBinding" bindingConfiguration="DefaultSoap"
       contract="Default.DefaultSoap" name="DefaultSoap" />
    </client>
  </system.serviceModel>
</configuration>
```

4. Rebuild the project.

Related Links

To Configure the Client Application

Copyright 2019 MYOB Technology Pty Ltd

Example 1.1.2: Configuring the Client Application (Screen-Based SOAP)

In this example, you will use the WSDL description of the *MYBIINTEGRATION* web service, which has been preconfigured for this training course on the Web Services (SM207040) form. This web service includes the WSDL descriptions of the forms that are used in this course. (Configuration of the web service is outside of the scope of this course. For details about the configuration, see *Additional Information: WSDL File of the Screen-Based Web Service*.)

For this web service, you will import the WSDL description into a Visual Studio project to make the objects and methods exposed by the WSDL description available for the integration application.

You will use the Web Services form to obtain the WSDL description of the web service.

Importing the WSDL File into a Visual Studio Project

To import the WSDL description into your development environment, proceed as follows:

Instead of creating a Visual Studio project, you can use the solution provided with this course (`MyBIIntegration\SBAPI\MyBIIntegrationSBAPI.sln`). This solution is configured for this course and already contains all the methods that are used in this course. You can use this solution for testing.

1. Open Visual Studio, and create a new Visual C# console application with the *MyBIIntegrationSBAPI* name and the *.NET Framework 4.7.1* target framework.

   To create a console application, click **File** > **New** > **Project** on the main menu of Visual Studio. In the **New Project** dialog box, which opens, do the following:

   1. Select the *Console App (.Net Framework)* template in the Visual C# installed templates.
   2. Specify the name, target framework, and location of the application.
   3. Click **OK**.

   The MyBIIntegration application that you have created contains the `Program.cs` file with the `Program` class.

2. Add to the project a reference to the MYOB Advanced ERP web service as follows:
   a. On the main menu of Visual Studio, click **Project** > **Add Service Reference**.
   b. In the **Add Service Reference** dialog box, which appears, click **Advanced**.
   c. In the **Service Reference Settings** dialog box, which appears, click **Add Web Reference**.
   d. In the **URL** box of the **Add Web Reference** dialog box, which appears, specify the URL of the *MYBIINTEGRATION* web service (see item 1 in the following screenshot).

      To copy the URL of the service, on the Web Services (SM207040) form, select *MYBIINTEGRATION* in the **Service ID** box, click **View Generated** on the form toolbar, and copy the URL from the address line in the browser for the page that opens. In this example, the URL is *http://localhost/MyStoreInstance/Soap/MYBIINTEGRATION.asmx*.

   e. Click **Go** (2) to make Visual Studio connect to the web service.
   f. In the **Web reference name** box, type `MyBIIntegration` (3). This name will be used as a namespace for the web service classes generated by Visual Studio based on the WSDL description of the service.
   g. Click **Add Reference** (4) to close the dialog box and add to the project the reference to the specified service.

Copyright 2019 MYOB Technology Pty Ltd

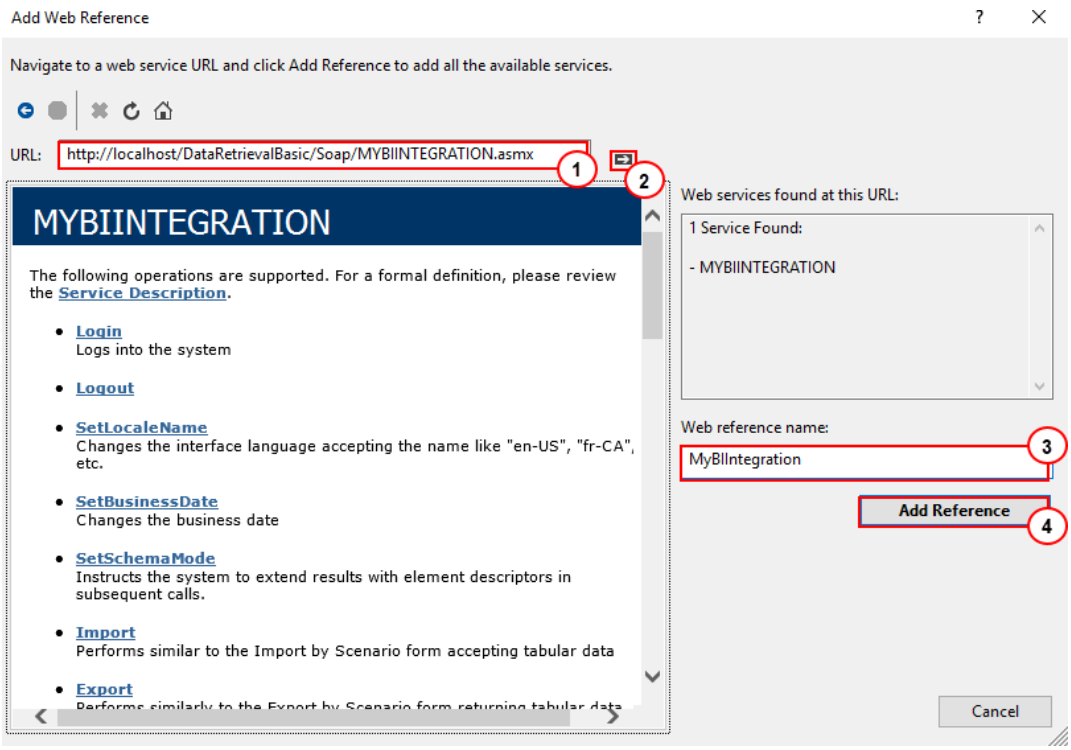


Figure: Add Web Reference dialog box

Visual Studio adds to the project the `Web References` project folder with the *MyBIIntegration* web service reference in it, as shown in the following screenshot.

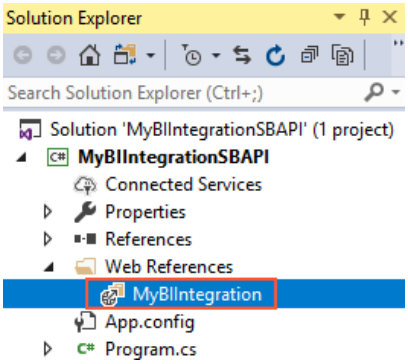


Figure: Solution Explorer

3. Rebuild the project.

Related Links

To Import the WSDL File Into the Development Environment

Copyright 2019 MYOB Technology Pty Ltd

Additional Information: CORS Support for OData

The information in this topic applies only to the OData interface.

The scenario described in this topic is outside of the scope of this course but may be useful to some readers.

MYOB Advanced ERP supports cross-origin resource sharing (CORS), meaning that requests for resources can come from a different domain than that of the resource making the request. With CORS enabled, you can allow access to the OData interface of your MYOB Advanced ERP instance for the client applications.

For more information about the CORS support, see *Configuring CORS* in *OData Support*.

For details about how to configure CORS for your instance of MYOB Advanced ERP, see *To Configure CORS*.

Copyright 2019 MYOB Technology Pty Ltd

Additional Information: Custom Endpoints and Endpoint Extensions

The information in this topic applies only to the contract-based REST and SOAP APIs.

The scenario described in this topic is outside of the scope of this course but may be useful to some readers.

The API provided by the system endpoints of MYOB Advanced ERP may not be sufficient for the requirements of your application. For example, you may need to retrieve data from a custom generic inquiry, or
you may have added new data access classes and business logic to MYOB Advanced ERP in a customization project. If the entity from which you want to retrieve data is not available in any of the system endpoints of MYOB Advanced ERP, to retrieve the data from this entity, you should create a custom endpoint either from scratch or by extending an existing endpoint.

For details about custom endpoints and endpoint extensions, see *Custom Endpoints and Endpoint Extensions* in the documentation.

Copyright 2019 MYOB Technology Pty Ltd

Lesson Summary

In this lesson, you have learned how to configure a client application to work with MYOB Advanced ERP contract-based SOAP and screen-based SOAP web services. For the contract-based SOAP web service, you have added a reference to the web service to the project and configured the application to use cookies. For the screen-based SOAP web service, you have added a reference to the generated service to the project.

You have also learned that the integration applications that use the OData protocol or the contract-based REST API do not require any specific configuration.

In addition, you have reviewed the possible options to configure MYOB Advanced ERP for integration.

The following table summarizes the configuration of MYOB Advanced ERP and the integration application that should be performed for each of the integration interfaces.

Integration Interface	Configuration of MYOB Advanced ERP	Configuration of the Integration Application
OData interface	Optional: Configuration of CORS	No
REST API	Optional: Configuration of a custom endpoint or an endpoint extension	No
Contract-based SOAP API	Optional: Configuration of a custom endpoint or endpoint extension	• Import of the WSDL description of the web service • Configuration of the integration application to use cookies
Screen-based SOAP API	Optional: Configuration of the WSDL description of the web service	Import of the WSDL description of the web service

Copyright 2019 MYOB Technology Pty Ltd

Lesson 1.2: Signing In to and Signing Out from MYOB Advanced ERP

Before you can retrieve data from MYOB Advanced ERP by using the integration interfaces provided by MYOB Advanced ERP, you need to sign in to MYOB Advanced ERP. In this lesson, you will sign in for different integration interfaces as follows:

- For the integration application that uses the OData protocol, you will use basic authentication with a username and password.
- In the REST application, you will add the requests for signing in to MYOB Advanced ERP using the sign- in endpoint.
- In the SOAP applications, you will add the code that signs in to MYOB Advanced ERP using the sign-in API methods.

For the REST application and SOAP applications, you will also add the request or the code that signs out from MYOB Advanced ERP.

For the contract-based REST and SOAP applications, you can also use the OAuth 2.0 authorization of applications, which is outside of the scope of this course. For a brief description, see *Additional Information: OAuth 2.0 Authorization*.

Lesson Objectives

In this lesson, you will learn how to:

- Sign in to MYOB Advanced ERP to use the OData interface.
- Sign in to MYOB Advanced ERP through the web services APIs.
- Sign out from MYOB Advanced ERP through the web services APIs.

Copyright 2019 MYOB Technology Pty Ltd

Example 1.2.1: Using OData

The OData protocol uses the basic authentication in MYOB Advanced ERP. That is, you need to sign in with a username and password to MYOB Advanced ERP before you request the data through the OData protocol.

In this example, you will configure a Postman collection to use the basic authentication and retrieve the list of generic inquiries available through OData to test the sign-in. If you want to use the browser to test the OData request, you can skip the configuration of a Postman collection, and enter your username and password in the browser when you are asked to do so. When the session expires, you have to sign in once again.

A sign-in through the OData protocol is treated by the system as a sign-in of a conventional user (not an API user). The license restriction for conventional users is shown in the **Concurrent Users** box on the **License** tab of the License Monitoring Console (SM604000) form.

Testing the Sign-In to MYOB Advanced ERP

To configure basic authorization in Postman and test it, do the following:

1. In Postman, create a new collection and configure its authorization settings (on the **Authorization** tab of the **Edit Collection** dialog box) as follows:
   a. In the **Type** box, select the *Basic Auth* type.
   b. In the **Username** and **Password** boxes, type the username and password that you are using to access the MYOB Advanced ERP instance for the training course.
   c. Click **Update**.
2. In the collection, add a `GET` request to the following URL to retrieve the list of generic inquiries exposed through OData.

```
http://localhost/MyStoreInstance/OData
```

The MYOB Advanced ERP instance that you are using for the training course has only one tenant configured. Therefore, you should not specify the tenant name in the URL. If the instance that you are accessing through the OData protocol has multiple tenants configured, you have to specify the tenant name in the URL. For example, if you need to retrieve the list of generic inquiries exposed through OData in the *MyTenant* tenant in the *MyStoreInstance* instance, you would use the following URL.

```
http://localhost/MyStoreInstance/OData/MyTenant
```

3. Send the request. The response of the successful request contains the `200 OK` status code. The following code example shows the response body.

```
<?xml version="1.0" encoding="utf-8"?>
<service xml:base="http://localhost/MyStoreInstance/odata/"
  xmlns="http://www.w3.org/2007/app" xmlns:atom="http://www.w3.org/2005/Atom">
    <workspace>
        <atom:title type="text">Default</atom:title>
        <collection href="BI-Employees">
            <atom:title type="text">BI-Employees</atom:title>
        </collection>
        <collection href="CR-CampaignSalesOrders">
            <atom:title type="text">CR-CampaignSalesOrders</atom:title>
        </collection>
        <collection href="BI-Dates">
            <atom:title type="text">BI-Dates</atom:title>
        </collection>
```

Copyright 2019 MYOB Technology Pty Ltd

```
        <collection href="BI-Opportunities">
            <atom:title type="text">BI-Opportunities</atom:title>
        </collection>
        <collection href="CR-CampaignInvoices">
            <atom:title type="text">CR-CampaignInvoices</atom:title>
        </collection>
        <collection href="BI-Customers">
            <atom:title type="text">BI-Customers</atom:title>
        </collection>
        <collection href="Item%20Availability%20Data">
            <atom:title type="text">Item Availability Data</atom:title>
        </collection>
        <collection href="Modified%20Stock%20Items">
            <atom:title type="text">Modified Stock Items</atom:title>
        </collection>
        <collection href="Stock%20Item%20Attachments">
            <atom:title type="text">Stock Item Attachments</atom:title>
        </collection>
        <collection href="Customer%20Contacts">
            <atom:title type="text">Customer Contacts</atom:title>
        </collection>
    </workspace>
</service>
```

4. Save the request.

Related Links

OData Support
License Restrictions for the Number of MYOB Advanced ERP Users
Support of Multiple Tenants

Copyright 2019 MYOB Technology Pty Ltd

Example 1.2.2: Using the REST API

In this example, through the REST API, you will sign in to and sign out from MYOB Advanced ERP.

To sign in to MYOB Advanced ERP, you will use the `POST` HTTP method and the endpoint for signing in. You will pass the authentication parameters (the username, password, tenant name, and company ID) in the body of the request.

With each subsequent request to the service, the application has to pass the cookies that it has received during sign-in. The Postman application, which you use for testing the REST API requests in this training course, appends the cookies to subsequent requests automatically.

To sign out from MYOB Advanced ERP, you will use the `POST` HTTP method and the endpoint for signing out.

For each sign-in to MYOB Advanced ERP, you must sign out after you finish your work with MYOB Advanced ERP to close the session. If the session is not closed, you may have issues with subsequent sign-ins to MYOB Advanced ERP through the web service APIs. The trial license, which you are using in the instance for this training course, limits the number of API users to two, as you can see in the **Maximum Number of Web Services API Users** box on the **License** tab of the License Monitoring Console (SM604000) form. Thus, if the session is not closed twice, you will not be able to sign in to MYOB Advanced ERP through the web service APIs for ten minutes. For details on how you can deal with this issue, see *Appendix: Troubleshooting*.

Signing In to MYOB Advanced ERP

To sign in to MYOB Advanced ERP, do the following:

1. In Postman, configure the following settings of the contract-based REST API request:
   - HTTP method: `POST`
   - URL: *http://localhost/MyStoreInstance/entity/auth/login*
   - Headers:

Key	Value
`Accept`	`application/json`
`Content-Type`	`application/json`

   - Body of the request:

```
{
  "name" : "andrwes",
  "password" : "myob1234"
}
```

     You specify the values in the body of a sign-in request as follows:
     - In the `name` element, you specify the username that the application should use to sign in to MYOB Advanced ERP.
     - In the `password` element, you specify the password for the username.
2. Send the request. If the request is successful, the response contains the `204 No Content` status code and the cookies that should be sent with subsequent requests to MYOB Advanced ERP. In Postman, you can view the cookies on the **Cookies** tab, which is shown in the following screenshot. The list of cookies contains five elements, one of which is `.ASPXAUTH`.
3.

Copyright 2019 MYOB Technology Pty Ltd

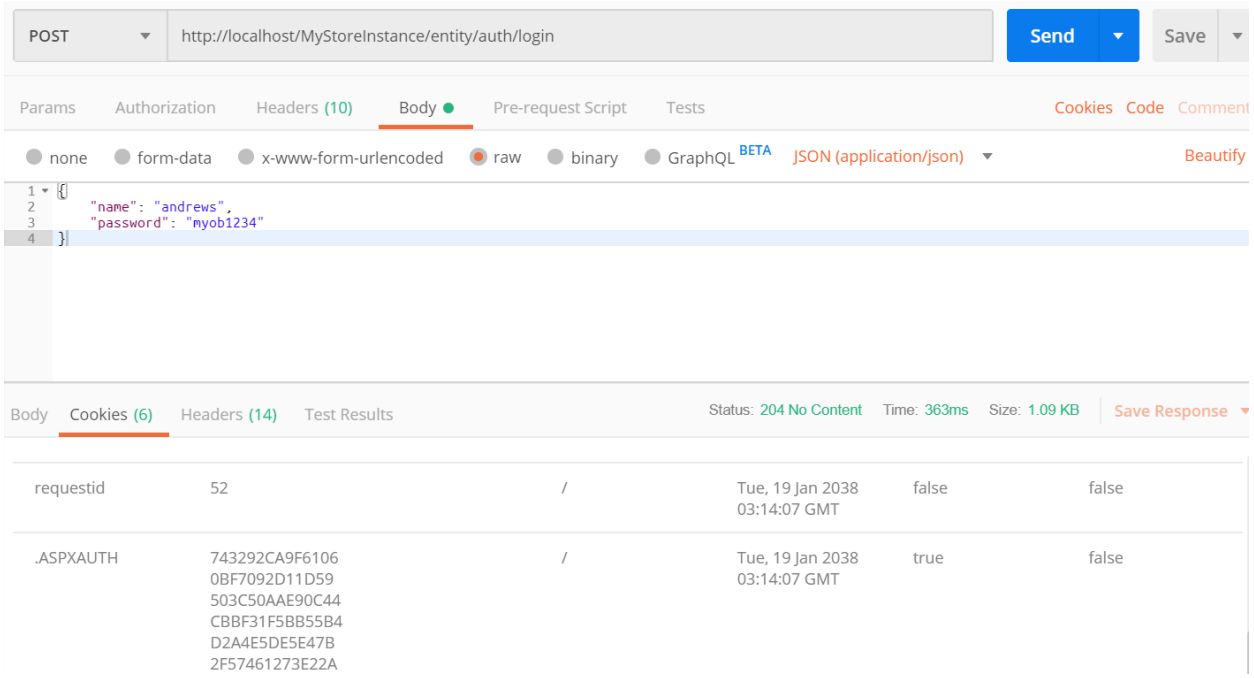


Figure: Cookies after signing in

4. Save the request.

Signing Out from MYOB Advanced ERP

To sign out from MYOB Advanced ERP, do the following:

1. In Postman, configure the contract-based REST API request as follows:
   - HTTP method: `POST`
   - URL: *http://localhost/MyStoreInstance/entity/auth/logout*
   - Headers:

Key	Value
`Accept`	`application/json`
`Content-Type`	`application/json`

2. Send the request. The response of the successful request contains the `204 No Content` status code and the list of cookies, which now contains four elements (the `.ASPXAUTH` element is missing), as shown in the following screenshot.

Copyright 2019 MYOB Technology Pty Ltd

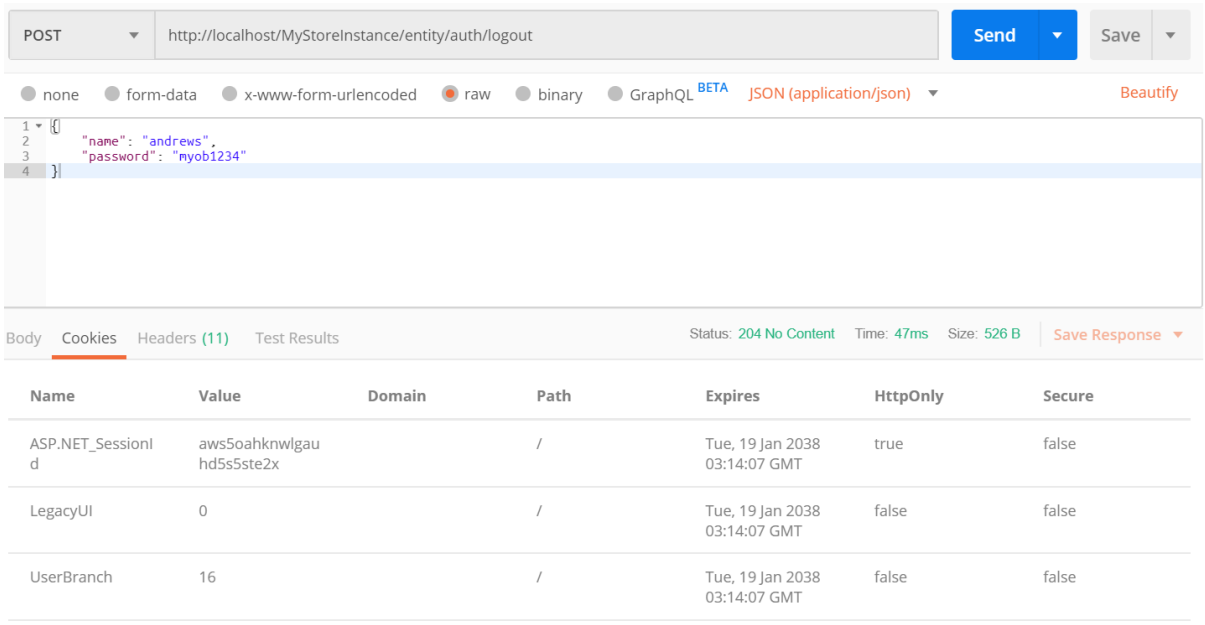


Figure: Cookies after signing out

3. Save the request.

Related Links

Login to the Service
License Restrictions for the Number of MYOB Advanced ERP Users
Support of Multiple Tenants
Companies and Branches
Logout from the Service

Copyright 2019 MYOB Technology Pty Ltd

Example 1.2.3: Using the Contract-Based SOAP API

In this example, you will specify the parameters of connection with the MYOB Advanced ERP contract-based SOAP web service. You will also add the code that signs in to and signs out from MYOB Advanced ERP.

To sign in to MYOB Advanced ERP, you will use the `Login()` method of a `DefaultSoapClient` object with the parameters (the username, password, tenant name, and company ID) that are specified in the application settings.

With each subsequent request to the service, the application has to pass the cookies that the application has received during sign-in. In *Example 1.1.1. Configuring the Client Application (Contract-Based SOAP)*, you have configured the application to send cookies with each API call to the service.

To sign out from MYOB Advanced ERP, you will use the `Logout()` method of a `DefaultSoapClient` object.

For each sign-in to MYOB Advanced ERP, you must sign out after you finish your work with MYOB Advanced ERP to close the session. If the session is not closed, you may have issues with subsequent sign-ins to MYOB Advanced ERP through the web service APIs. The trial license, which you are using in the instance for this training course, limits the number of API users to two, as you can see in the **Maximum Number of Web Services API Users** box on the **License** tab of the License Monitoring Console (SM604000) form. Thus, if the session is not closed twice, you will not be able to sign in to MYOB Advanced ERP through the web service APIs for ten minutes. For details on how you can deal with this issue, see *Appendix: Troubleshooting*.

For each call of the `Login()` method, you must call the `Logout()` method. Thus, we recommend that you call the `Logout()` method in the `finally` block.

Signing In to and Signing Out from MYOB Advanced ERP

Proceed as follows:

1. Configure the settings of the *MyBIIntegration* project as follows:
    a. Right-click the **MyBIIntegration** project folder in Solution Explorer and select **Properties** from the context menu. On the **MyBIIntegration** tab, which opens, click **Settings** and then **This project does not contain a default settings file. Click here to create one.** Visual Studio creates the `Settings.settings` file in the `Properties` folder of the *MyBIIntegration* project and opens the file.
    b. Add to the application settings the `Username`, `Tenant`, `Company`, and `Password` properties, which are shown in the following screenshot. Specify the values of the properties in the **Value** column as follows:
        - As the *Username*, specify the username (in this example, `admin`) that the application should use to sign in to MYOB Advanced ERP.
        - As the *Tenant*, specify the name of the tenant to which the application should sign in. This is the name (*MyStore*) that you have specified during the creation of the MYOB Advanced ERP instance for the training course. You can view the name that should be used for the tenant in the **Login Name** box of the Tenants (SM203520) form.
        - As the *Company*, specify the ID of the company (*MYSTORE*) to which the application should sign in. You can view the ID of the company in the **Company ID** column of the Companies (CS1015PL) form.
        - As the *Password*, specify the password for the username.

Copyright 2019 MYOB Technology Pty Ltd

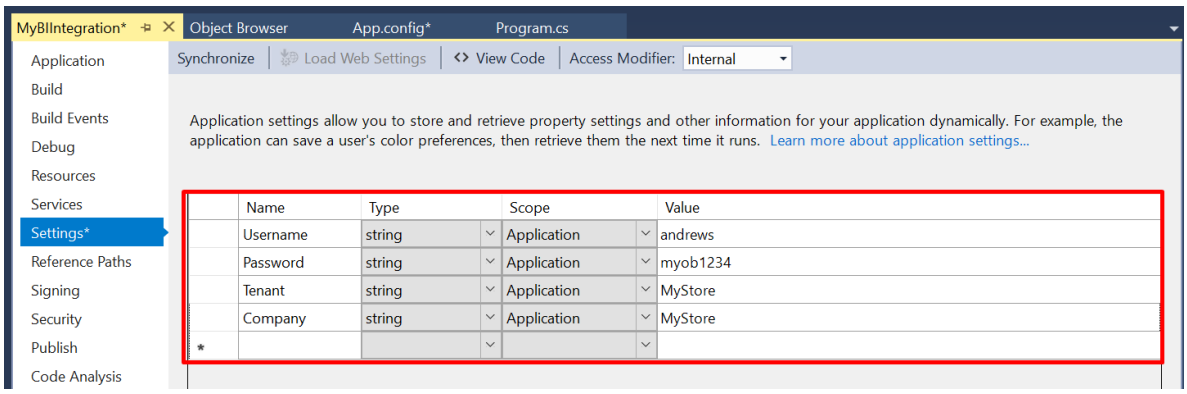


Figure: Application settings

2. In the `Main()` method of the `Program` class, add the code marked in bold type below.

```
...
using MyBIIntegration.Default;

namespace MyBIIntegration
{
  class Program
  {
    static void Main(string[] args)
    {
      //Using the Default/18.200.001 endpoint
      using (Default.DefaultSoapClient soapClient =
           new Default.DefaultSoapClient())
        {
            //Sign in to MYOB Advanced ERP
            soapClient.Login
            (
                Properties.Settings.Default.Username,
                Properties.Settings.Default.Password,
                Properties.Settings.Default.Tenant,
                Properties.Settings.Default.Company,
                null
            );

            try
            {
                //You will add the integration code here
            }
            catch (Exception e)
            {
                Console.WriteLine(e);
                Console.WriteLine();
                Console.WriteLine("Press any key to continue");
                Console.ReadLine();
            }
            finally
            {
                //Sign out from MYOB Advanced ERP
                soapClient.Logout();
            }
        }
    }
```

Copyright 2019 MYOB Technology Pty Ltd

```
    }
}
```

In this code, you have added the `MyBIIntegration.Default using` directive to make the `Program`class use the *Default* service reference and to be able to use the classes and methods of the contract-based SOAP API.

3. Rebuild the project.

Related Links

Login() Method
License Restrictions for the Number of MYOB Advanced ERP Users
Support of Multiple Tenants
Companies and Branches
Logout() Method

Copyright 2019 MYOB Technology Pty Ltd

Example 1.2.4: Using the Screen-Based SOAP API

In this example, you will specify the parameters of connection with the MYOB Advanced ERP screen-based SOAP web service. You will also add the code that signs in to and signs out from MYOB Advanced ERP.

You will create a `Screen` object, which is the main object that provides access to all other objects and methods of the screen-based SOAP API. Before the sign-in to MYOB Advanced ERP, you will initialize the `CookieContainer` property of the object with a new `System.Net.CookieContainer` object, which maintains the session state for a client. You will also specify the URL of the web service in the `URL` property of the object. This is the same URL that you specified when you added a web reference to the Visual Studio project.

To sign in to MYOB Advanced ERP, you will use the `Login()` method of the `Screen` object with the parameters (the username and password) that are specified in the application settings. You will specify the username in the following format: `Username@TenantName:CompanyName`.

To sign out from MYOB Advanced ERP, you will use the `Logout()` method of the `Screen` object.

For each sign-in to MYOB Advanced ERP, you must sign out after you finish your work with MYOB Advanced ERP to close the session. If the session is not closed, you may have issues with subsequent sign-ins to MYOB Advanced ERP through the web service APIs. The trial license, which you are using in the instance for this training course, limits the number of API users to two, as you can see in the **Maximum Number of Web Services API Users** box on the **License** tab of the License Monitoring Console (SM604000) form. Thus, if the session is not closed twice, you will not be able to sign in to MYOB Advanced ERP through the web service APIs for ten minutes. For details on how you can deal with this issue, see *Appendix: Troubleshooting*.

For each call of the `Login()` method, you must call the `Logout()` method. Thus, we recommend that you call the `Logout()` method in the `finally` block.

Instructions for Signing In to and Signing Out from MYOB Advanced ERP

Proceed as follows:

1. In the `Properties` folder of the *MyBIIntegrationSBAPI* project, double-click `Settings.settings`, and add to the application settings the `Username` and `Password` properties, which are shown in the following screenshot. Specify the values of the properties in the **Value** column as follows:

   - As the *Username*, specify the username that the application should use to sign in to MYOB Advanced ERP in the following format: `Username@TenantName:CompanyName`.

     The tenant name (in this example, *MyStore*) is the name that you have specified during the creation of the MYOB Advanced ERP instance for the training course. You can view the name that should be used for the tenant in the **Login Name** box of the Tenants (SM203520) form.

     For this training course, you use the *MYSTORE* company. You can view the ID of the company in the **Company ID** column of the Companies (CS1015PL) form.

     That is, in this example, you should specify the username as follows: `andrews@MyStore:MYSTORE`.

   - As the *Password*, you specify the password for the username.

   The *MyBIIntegrationSBAPI_MyBIIntegration_Screen* setting, which Visual Studio added to the application settings when you added the web reference to the project, contains the URL of the web service.

Copyright 2019 MYOB Technology Pty Ltd

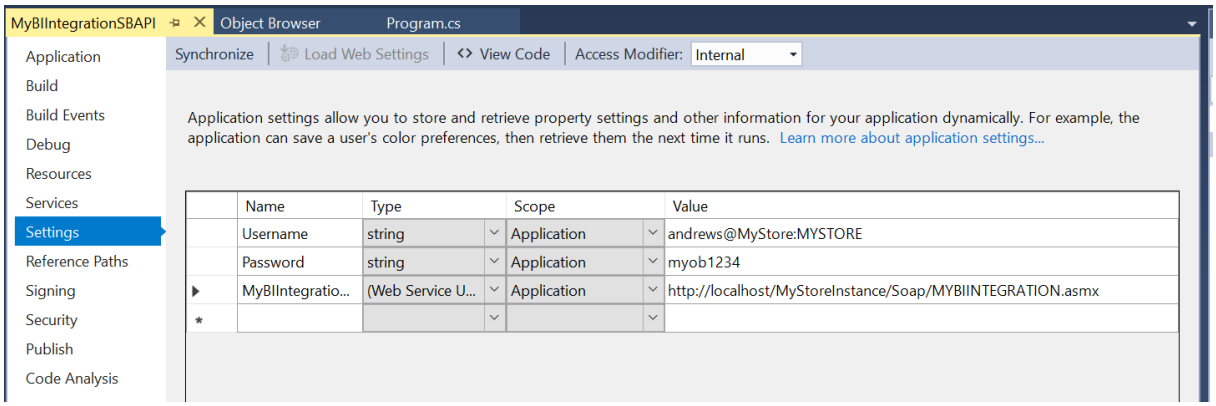


Figure: Application settings

2 In the `Main()` method of the `Program` class, add the code marked in bold type below.

```
...
using MyBIIntegrationSBAPI.MyBIIntegration;

namespace MyBIIntegrationSBAPI
{
    class Program
    {
        static void Main(string[] args)
        {
            using (Screen screen = new Screen())
            {
                //Specify the connection parameters
                screen.CookieContainer = new System.Net.CookieContainer();
                screen.Url = Properties.Settings.
                  Default.MyBIIntegrationSBAPI_MyBIIntegration_Screen;

                //Sign in to MYOB Advanced ERP
                screen.Login
                (
                    Properties.Settings.Default.Username,
                    Properties.Settings.Default.Password
                );

                try
                {
                    //You will add the integration code here
                }
                catch (Exception e)
                {
                    Console.WriteLine(e);
                    Console.WriteLine();
                    Console.WriteLine("Press any key to continue");
                    Console.ReadLine();
                }
                finally
                {
                    //Sign out from MYOB Advanced ERP
                    screen.Logout();
                }
```

Copyright 2019 MYOB Technology Pty Ltd

```
                }
            }
        }
    }
```

In this code, you have added the `MyBIIntegrationSBAPI.MyBIIntegration using` directive to make the `Program`class use the *MyBIIntegration* service reference and to be able to use the classes and methods of the screen-based SOAP API.

3. Rebuild the project.

Related Links

Login() Method
License Restrictions for the Number of MYOB Advanced ERP Users
Support of Multiple Tenants
Companies and Branches
Logout() Method

Copyright 2019 MYOB Technology Pty Ltd

Additional Information: OAuth 2.0 Authorization

The information in this topic applies only to the contract-based APIs. The OAuth 2.0 authorization is not supported for the OData interface and the screen-based SOAP API in MYOB Advanced ERP.

The scenario described in this topic is outside of the scope of this course but may be useful to some readers.

MYOB Advanced ERP supports the OAuth 2.0 mechanism of authorization for applications that are integrated with MYOB Advanced ERP through APIs. When a client application of MYOB Advanced ERP uses the OAuth 2.0 mechanism of authorization, the client application does not operate with the MYOB Advanced ERP credentials to sign in a user to MYOB Advanced ERP; instead, the application obtains an access token from MYOB Advanced ERP and uses this token when it requests data from MYOB Advanced ERP.

For details about the OAuth 2.0 authorization mechanism, see *Authorizing Client Applications to Work with MYOB Advanced ERP*. The authorization of a client application to work with MYOB Advanced ERP is also illustrated in the *I310 Web Services: Advanced* | *Data Retrieval* training course.

Copyright 2019 MYOB Technology Pty Ltd

Lesson Summary

In this lesson, you have learned how an application that uses the OData protocol can sign in to MYOB Advanced ERP. You have also learned how to sign in to and sign out from MYOB Advanced ERP through the web services APIs.

You have also reviewed the possible options to configure MYOB Advanced ERP for integration.

The following table summarizes the sign-in and sign-out options and limitations for each of the integration interfaces.

Integration Interface	Authentication and Authorization	Limit for the Number of API Users
OData interface	Basic authentication	No
REST API	• `POST` requests to the sign-in and sign-out endpoints • OAuth 2.0 authorization	Yes
Contract-based SOAP API	• The `Login()` and `Logout()` of a `DefaultSoapClient` object • OAuth 2.0 authorization	Yes
Screen-based SOAP API	The `Login()` and `Logout()` of a `Screen` object	Yes

Copyright 2019 MYOB Technology Pty Ltd

Part 2: Initial Data Retrieval

In the BI application of the MyStore company, a marketing manager should be able to view analytics for the distribution of existing customers by geographical state. To display to the marketing manager the information about the customers, the MyBIIntegration application needs to retrieve from MYOB Advanced ERP the information on the customers, their contacts, and their addresses.

Also, in the BI application of the MyStore company, a warehouse manager should be able to view the statistics of the items available in the warehouse. To make it possible for the BI application to display these statistics, the MyBIIntegration solution needs to retrieve from MYOB Advanced ERP the information about the quantities of stock items available in the warehouses.

By completing the examples of this part, you will configure the MyBIIntegration application to export the full list of customer records with contacts and addresses. (The import of this list from the MyBIIntegration application to the BI application is outside of the scope of this course.) You will also configure the MyBIIntegration application to export the quantities of stock items. You can use all of the available integration interfaces for exporting this data.

Copyright 2019 MYOB Technology Pty Ltd

Lesson 2.1: Retrieving the List of Customers with Contacts

The BI application of the MyStore company should display to a user the distribution of existing customers by geographic state. The data of these customers has been entered on the Customers (AR303000) form in MYOB Advanced ERP. To provide the list of customers to the BI application, the MyBIIntegration application should export the following values from MYOB Advanced ERP (with the corresponding locations on the Customers form):

- Customer ID (the **Customer ID** box of the Summary area)
- Customer name (the **Customer Name** box of the Summary area)
- Customer class (the **Customer Class** box of the **Financial Settings** section on the **General Info** tab)
- Details of the main contact of the customer (the **Main Contact** section of the **General Info** tab):
    - Email address (the **Email** box)
    - Primary phone number (the **Phone 1** box)
- Customer address (the **Main Address** section of the **General Info** tab):
    - City (the **City** box)
    - State (the **State** box)
    - Postal code (the **Postal Code** box)
    - Address line (the **Address Line 1** and **Address Line 2** boxes)

These elements are shown in the following screenshot.

Copyright 2019 MYOB Technology Pty Ltd

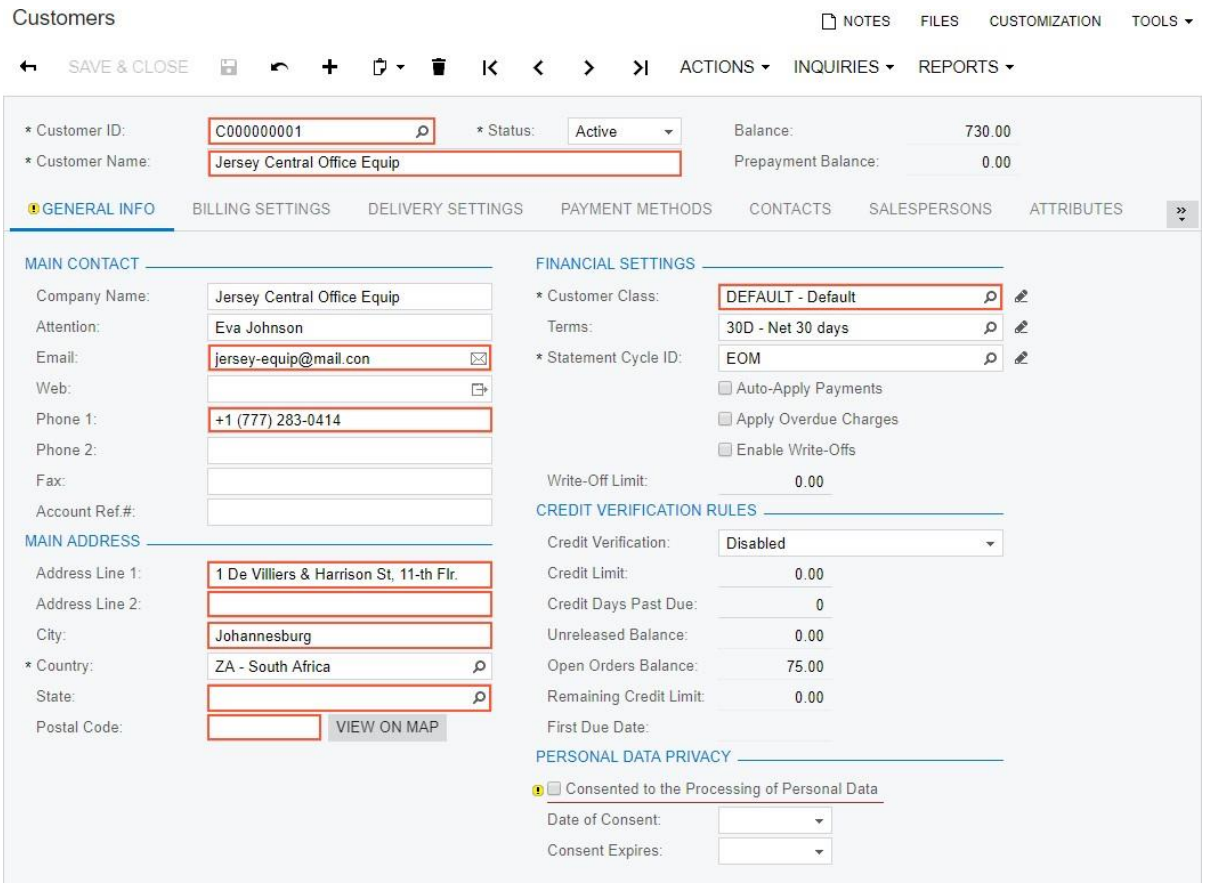


Figure: Elements on the Customers form

This lesson shows how you can implement this scenario by using the OData interface, contract-based REST API, contract-based SOAP API, and screen-based SOAP API. To export this data by using the OData protocol, you need to create a generic inquiry. Through the web services APIs, you can export this data from MYOB Advanced ERP without any additional preparations.

Lesson Objective

In this lesson, you will learn how to retrieve the data from an MYOB Advanced ERP data entry form.

Copyright 2019 MYOB Technology Pty Ltd

Example 2.1.1: Using a Custom Generic Inquiry (OData)

In this example, through the OData protocol, you will export the list of customer records with contacts.

Through the OData protocol, you cannot export records directly from a data entry form, such as the Customers (AR303000) form. Thus, before the export, you have to configure a generic inquiry that retrieves the needed data from MYOB Advanced ERP.

In this example, you will use the Customer Contacts custom generic inquiry, which has been preconfigured for this training course. This generic inquiry has no parameters and is based on the `PX.Objects.AR.Customer`, `PX.Objects.CR.Address`, and `PX.Objects.CR.Contact` database tables. You can view this generic inquiry on the Generic Inquiry (SM208000) form by selecting the inquiry with the title *Customer Contacts* in the Summary area and clicking **View Inquiry** on the form toolbar.

In the training data, the **Expose via OData** check box is selected for this generic inquiry in the Summary area of the Generic Inquiry form. This means that the generic inquiry is available through the OData interface.

You will obtain the results of the generic inquiry in JSON format by using the `$format` parameter of the request.

Prerequisites

Before you send the request in the example, you need to sign in to MYOB Advanced ERP as described in *Example 1.2.1: Using OData*. If you have created a Postman collection with basic authentication configured, add the request described below to the collection and configure it to inherit the authorization type from the parent collection.

Retrieving the List of Customers

To retrieve the list of customers with contacts, do the following:

1. Configure the following settings of the request:
   - HTTP method: `GET`
   - URL: *http://localhost/MyStoreInstance/OData/Customer%20Contacts*
   - Parameter: `$format=json`
2. Send the request. If the request is successful, its response contains the `200 OK` status code. The following code example shows a fragment of the response body.

```
{
  "odata.metadata":
    "http://localhost/MyStoreInstance/odata/$metadata#Customer%20Contacts",
  "value":
  [
    {
      "CustomerID": "C000000001",
      "CustomerName": "Jersey Central Office Equip",
      "CustomerClass": "DEFAULT",
      "Email": "jersey-equip@mail.con",
      "Phone1": "+1 (777) 283-0414",
      "City": "Johannesburg",
      "State": null,
      "PostalCode": null,
      "AddressLine1": "1 De Villiers & Harrison St, 11-th Flr.",
      "AddressLine2": null
    },
```

Copyright 2019 MYOB Technology Pty Ltd

```
      ...
   ]
}
```

3. Save the request.

Related Links

OData Support

Copyright 2019 MYOB Technology Pty Ltd

Example 2.1.2: Using GET and the $expand and $select Parameters (REST)

In this example, through the REST API, you will configure an HTTP request that exports the list of customer records with contacts and addresses.

You will use the `GET` HTTP method and the `Customer` entity of the *Default/18.200.001* endpoint to list customer records. The `Customer` entity is mapped to the Customers (AR303000) form.

Because the database can contain thousands of customer records and each record includes dozens of fields, to achieve the best performance of the application, you need to specify the fields of the customer records that should be returned. You will use the `$select` parameter to specify the fields whose values should be retrieved from MYOB Advanced ERP for each customer record. For a customer record, you will export the values of the following fields of the `Customer` entity:

- `CustomerID`
- `CustomerName`
- `CustomerClass`
- `MainContact` (nested `Contact` entity):
  - `Email`
  - `Phone1`
  - `Address` (nested `Address` entity):
    - `AddressLine1`
    - `AddressLine2`
    - `City`
    - `State`
    - `PostalCode`

By using the approach described in this example, you can export the values of the fields that are available in the endpoint. To export any other fields (such as the fields added to an MYOB Advanced ERP form in a customization project), you should use the approach described in *Additional Information: Retrieval of Custom Fields*.

You will use the `$expand` parameter to specify the nested entities to be returned. In this parameter, you have to specify all the nested entities whose fields you need to retrieve from MYOB Advanced ERP. That is, in this example, you have to specify the following entities: `MainContact` and `MainContact/Address`.

Prerequisites

Before you can test the example, you need to sign in to MYOB Advanced ERP as described in *Example 1.2.2: Using the REST API*. For the request in this example, you have to pass the cookies that you have received during the sign-in.

Instructions for Retrieving the List of Customers

To retrieve the list of customers with contacts, do the following:

1. In Postman, configure the following settings of the contract-based REST API request:
   - HTTP method: `GET`
   - URL: *http://localhost/MyStoreInstance/entity/Default/18.200.001/Customer*
   - Parameters:

Copyright 2019 MYOB Technology Pty Ltd

Parameter	Value
`$expand`	`MainContact,MainContact/Address`
`$select`	`CustomerID,CustomerName,CustomerClass,MainContact/Email,MainContact/Phone1,MainContact/Address/AddressLine1,MainContact/Address/AddressLine2,MainContact/Address/City,MainContact/Address/State,MainContact/Address/PostalCode`

- Headers:

Key	Value
`Accept`	`application/json`
`Content-Type`	`application/json`

2. Send the request. If the request is successful, the response contains the `200 OK` status code and includes the list of requested fields of all customer records in JSON format. The following code example shows a fragment of the response body.

```
[
    {
        "id": "f475b40c-8f59-4750-b361-84a9fd34000d",
        "rowNumber": 1,
        "note": "",
        "CustomerClass": {
            "value": "DEFAULT"
        },
        "CustomerID": {
            "value": "C000000001"
        },
        "CustomerName": {
            "value": "Jersey Central Office Equip"
        },
        "MainContact": {
            "id": "d85c888a-d1c9-448a-a297-d3be2d6c9df9",
            "rowNumber": 1,
            "note": null,
            "Address": {
                "id": "6da0645e-63f1-4abc-85f3-a91f5308d817",
                "rowNumber": 1,
                "note": null,
                "AddressLine1": {
                    "value": "1 De Villiers & Harrison St, 11-th Flr."
                },
                "AddressLine2": {},
                "City": {
                    "value": "Johannesburg"
                },
                "PostalCode": {},
                "State": {},
                "custom": {},
                "files": []
            },
            "Email": {
                "value": "jersey-equip@mail.con"
            },
            "Phone1": {
                "value": "+1 (777) 283-0414"
            },
            "custom": {},
            "files": []
        },
        "custom": {},
```

Copyright 2019 MYOB Technology Pty Ltd

```
            "files": []
        },
        ...
]
```

3. Save the request.

Related Links

Retrieval of Records by Conditions
Parameters for Retrieving Records

Copyright 2019 MYOB Technology Pty Ltd

Example 2.1.3: Using the GetList() Method and the ReturnBehavior Property (Contract-Based SOAP)

In this example, you will add to the MyBIIntegration application the method that exports the list of customer records with contacts.

You will use the `Customer` class and the `GetList()` method of an instance of the `DefaultSoapClient` class available in the *Default* service reference. The `Customer` class corresponds to the `Customer` entity of the *Default/18.200.001* endpoint, which is mapped to the Customers (AR303000) form.

Because the database can contain thousands of customer records and each record includes dozens of fields, to achieve the best performance of the application, you need to specify the fields of the customer records that should be returned. You will specify the fields to be returned for each customer record by using the `StringReturn` objects. For a customer record, you will export the values of the following fields of the `Customer` entity:

- `CustomerID`
- `CustomerName`
- `CustomerClass`
- `MainContact` (nested `Contact` entity):
  - `Email`
  - `Phone1`
  - `Address` (nested `Address` entity):
    - `AddressLine1`
    - `AddressLine2`
    - `City`
    - `State`
    - `PostalCode`

By using the approach described in this example, you can export the values of the fields that are available in the endpoint. To export any other fields (such as the fields added to an MYOB Advanced ERP form in a customization project), you should use the approach described in *Additional Information: Retrieval of Custom Fields*.

You will set the `ReturnBehavior` property to `ReturnBehavior.OnlySpecified` for the `Customer` entity and each nested entity, to make the service return only the fields that you have specified by using the `StringReturn` objects.

Retrieving the List of Customers

To retrieve the list of customers with contacts, do the following:

1. Add the `Integration` folder to the *MyBIIntegration* project, and add a new C# class to it with the name `InitialDataRetrieval`.

   Visual Studio creates an `InitialDataRetrieval.cs` file in the *MyBIIntegration* project.

   To add a folder to the *MyBIIntegration* project, in **Solution Explorer**, right-click the *MyBIIntegration* project, select **Add** > **New Folder**, and specify the name of the folder to be created: `Integration`. To add the `InitialDataRetrieval` class to the folder, in **Solution Explorer**, right-click the folder, and select **Add** > **Class**. In the **Add New Item** dialog box, select the **Class** item, and type the name: `InitialDataRetrieval.cs`.

Copyright 2019 MYOB Technology Pty Ltd

2. In the `InitialDataRetrieval.cs` file, type the `using` directives as shown in the following code to make the `InitialDataRetrieval` class use the *Default* service reference and the `System.IO` classes.

```
using MyBIIntegration.Default;
using System.IO;
```

3. In the `InitialDataRetrieval` class, define the `RetrieveListOfCustomers()` method, as the following code shows.

```
//Retrieving the list of customers with contacts
public static void RetrieveListOfCustomers(
  DefaultSoapClient soapClient)
{
    Console.WriteLine("Retrieving the list of customers with contacts...");

    //Specify the parameters of the stock items to be returned
    Customer customersToBeRetrieved = new Customer
    {
        //Return the values of only the specified fields
        ReturnBehavior = ReturnBehavior.OnlySpecified,

        //Specify the other fields to be returned
        CustomerID = new StringReturn(),
        CustomerName = new StringReturn(),
        CustomerClass = new StringReturn(),
        MainContact = new Contact
        {
            ReturnBehavior = ReturnBehavior.OnlySpecified,
            Email = new StringReturn(),
            Phone1 = new StringReturn(),
            Address = new Address
            {
                ReturnBehavior = ReturnBehavior.OnlySpecified,
                AddressLine1 = new StringReturn(),
                AddressLine2 = new StringReturn(),
                City = new StringReturn(),
                State = new StringReturn(),
                PostalCode = new StringReturn()
            }
        }
    };

    //Get the list of customers
    Entity[] customers = soapClient.GetList(customersToBeRetrieved);

    //Save the results to a CSV file
    using (StreamWriter file = new StreamWriter("Customers.csv"))
    {
        //Add headers to the file
        file.WriteLine(
            "CustomerID;CustomerName;CustomerClass;Email;Phone1;" +
            "AddressLine1;AddressLine2;City;State;PostalCode;");

        //Write the values for each item
        foreach (Customer customer in customers)
        {
            file.WriteLine(
                string.Format("{0};{1};{2};{3};{4};{5};{6};{7};{8};{9};",
                customer.CustomerID.Value,
                customer.CustomerName.Value,
                customer.CustomerClass.Value,
                customer.MainContact.Email.Value,
                customer.MainContact.Phone1.Value,
                customer.MainContact.Address.AddressLine1.Value,
                customer.MainContact.Address.AddressLine2.Value,
                customer.MainContact.Address.City.Value,
                customer.MainContact.Address.State.Value,
```

Copyright 2019 MYOB Technology Pty Ltd

```
                    customer.MainContact.Address.PostalCode.Value));
            }
        }
    }
```

4 In the `try` block of the `Main()` method of the `Program` class, call the `RetrieveListOfCustomers()` method of the `InitialDataRetrieval` class, as the following code shows.

```
...
using MyBIIntegration.Integration;

namespace MyBIIntegration
{
  class Program
  {
    static void Main(string[] args)
    {
      //Using the Default/18.200.001  endpoint
      using (Default.DefaultSoapClient soapClient =
          new Default.DefaultSoapClient())
      {
        ...
        try
        {
          //Retrieving the list of customers with contacts
          InitialDataRetrieval.RetrieveListOfCustomers(soapClient);
        }
        ...
      }
    }
    ...
  }
}
```

5 Rebuild the project and run the application. If you run the project in Debug mode, you can find the `Customers.csv` file with the results of the export in the `\bin\Debug` folder of the *MyBIIntegration* project.

You can open the project folder in File Explorer by right-clicking the project in Solution Explorer and selecting **Open Folder in File Explorer**.

A fragment of the resulting file is shown in the following screenshot.

	A	B	C	D	E	F	G	H	I	J
1	CustomerID	CustomerName	CustomerClass	Email	Phone1	AddressLine1	AddressLine2	City	State	PostalCode
2	C000000001	Jersey Central Office Equip	DEFAULT	jersey-equip@mail.con	+1 (777) 283-0414	1 De Villiers & Harrison St, 11-th Flr.		Johannesburg		
3	C000000002	Microchip Restaurant	DEFAULT		+1 (777) 459-4255	1 Kalisa Way		Paramus	NJ	
4	C000000003	Jevy Computers	DEFAULT	info@jevy-comp.con	+1 (777) 380-0089	1000 Pennsylvania Ave		San Francisco	CA	94107-3479
5	C000000004	KRK Consulting Service	DEFAULT	consulting@krk.con	+1 (777) 829-6988	1 Penn Plz		New York	NY	10119
6	C000000005	Wright Corner	DEFAULT	mail@wright-corner.con	+1 (777) 513-9166	1 W 89TH St		New York	NY	10024
7	C000000006	NETCAFE NY	DEFAULT	ny@netvafe.con	+1 (777) 673-7392	1011 High Ridge Rd		Stamford	CT	
8	C000000007	Bestype Image	DEFAULT		+1 (777) 966-6886	105 Cecil St		Singapore		
9	C000000008	Digitech Printers	DEFAULT	Printers@Digitech.con	+1 (777) 623-6150	11 Chiswick Park		Cape town		
10	C000000009	Precision Photos	DEFAULT		+1 (777) 302-2756	11 US Air Terminal Ste 1413		Flushing	NY	

Figure: The `Customers.csv` file

Related Links

GetList() Method (Contract Version 3)
ReturnBehavior Property (Contract Version 3)

Copyright 2019 MYOB Technology Pty Ltd

Example 2.1.4: Using the Export() Method (Screen-Based SOAP)

In this example, you will add to the MyBIIntegration application the method that exports the list of customer records with contacts and addresses from the Customers (AR303000) form. The Customers form has the *AR303000* form ID. Therefore, to access this form, you will use the classes and methods with the *AR303000* prefix.

You will use the screen-based API wrapper to prevent application failures due to possible UI changes on the Customers form in later versions of MYOB Advanced ERP. To use the wrapper, you will add a reference to `PX.Soap.dll` to the project and obtain the schema of the Customers form by using the `GetSchema()` static method of the `PX.Soap.Helper` class. You will use this method instead of the `AR303000GetSchema()` method of a `Screen` object. The use of the screen-based API wrapper grants the application independence of the UI changes in MYOB Advanced ERP. With the `GetSchema()` method, you obtain a `Content` object that corresponds to the Customers (AR303000) form.

To specify the fields to be retrieved, you will configure an array of `Command` objects. In this array, you will add the `EveryCustomerID` service command, which is available through `ServiceCommands` of `CustomerSummary`, to make MYOB Advanced ERP export all customer records available on the form. In the array, you will also specify the fields by using the schema of the Customers form available in the `Content` object. The objects that are available in the `Content` object have names that are similar to the names of the UI elements that you see on the form.

You will use the `Export()` method of the `Screen` object to retrieve the values of the specified fields of the customer records. You will specify the parameters of the method as follows:

- Pass the configured array of commands as the first parameter.
- Because you need to retrieve the full list of customers, pass `null` as the second parameter (you do not need to apply any filter to the list), and pass `0` as the third parameter (you do not need to limit the number of records).
- Pass `true` as the fourth parameter, to include column headers in the result of export. The first row of the exported data will include the names of exported elements.
- Pass `false` as the fifth parameter, because you do not need to stop the export if an error occurs during this process.

Retrieving the List of Customers

To retrieve the list of customers with contacts, do the following:

1. Add the `Integration` folder to the *MyBIIntegrationSBAPI* project, and add a new C# class to it with the name `InitialDataRetrieval`.

   Visual Studio creates a `InitialDataRetrieval.cs` file in the *MyBIIntegrationSBAPI* project.

   To add a folder to the *MyBIIntegrationSBAPI* project, in **Solution Explorer**, right-click the *MyBIIntegrationSBAPI* project, select **Add** > **New Folder**, and specify the name of the folder to be created: `Integration`. To add the `InitialDataRetrieval` class to the folder, in **Solution Explorer**, right-click the folder, and select **Add** > **Class**. In the **Add New Item** dialog box, select the **Class** item, and type the name: `InitialDataRetrieval.cs`.

2. In the `InitialDataRetrieval.cs` file, type the `using` directives as shown in the following code to make the `InitialDataRetrieval` class use the *MyBIIntegration* web reference and the classes of the `System.IO` namespace.

```
using MyBIIntegrationSBAPI.MyBIIntegration;
using System.IO;
```

3. To use the screen-based API wrapper, add a reference to the `PX.Soap.dll` to the *MyBIIntegrationSBAPI* project. `PX.Soap.dll` is located in the `ScreenBasedAPIWrapper` folder

Copyright 2019 MYOB Technology Pty Ltd

in the MYOB Advanced ERP installation folder. (By default, it is `C:\Program Files\MYOB Advanced ERP \ScreenBasedAPIWrapper`).

4. In the `InitialDataRetrieval` class, define the `ExportCustomers()` method, as the following code shows.

```
public static void ExportCustomers(Screen screen)
{
    Console.WriteLine("Retrieving the list of customers with contacts...");

    //Get the schema of the Customers (AR303000) form and
    //configure the sequence of commands
    AR303000Content custSchema =
      PX.Soap.Helper.GetSchema<AR303000Content>(screen);
    var commands = new Command[]
    {
        //Get the values of the needed elements
        custSchema.CustomerSummary.ServiceCommands.EveryCustomerID,
        //Customer summary
        custSchema.CustomerSummary.CustomerID,
        custSchema.CustomerSummary.CustomerName,
        //General Info tab, Financial Settings
        custSchema.GeneralInfoFinancialSettings.CustomerClass,
        //General Info tab, Main Contact
        custSchema.GeneralInfoMainContact.Email,
        custSchema.GeneralInfoMainContact.Phone1,
        //General Info tab, Main Address
        custSchema.GeneralInfoMainAddress.AddressLine1,
        custSchema.GeneralInfoMainAddress.AddressLine2,
        custSchema.GeneralInfoMainAddress.City,
        custSchema.GeneralInfoMainAddress.State,
        custSchema.GeneralInfoMainAddress.PostalCode
    };

    //Export the customer data
    string[][] customerData =
      screen.AR303000Export(commands, null, 0, true, false);

    //Save the data to a CSV file
    StreamWriter file = new StreamWriter("Customers.csv");
    {
        foreach (string[] rows in customerData)
        {
            foreach (string row in rows)
            {
                file.Write(row + ";");
            }
            file.WriteLine();
        }
    }
    file.Close();
}
```

5. In the `try` block of the `Main()` method of the `Program` class, call the `ExportCustomers()` method of the `InitialDataRetrieval` class, as the following code shows.

```
...
using MyBIIntegrationSBAPI.Integration;

namespace MyBIIntegrationSBAPI
{
    class Program
    {
        static void Main(string[] args)
        {
            using (Screen screen = new Screen())
            {
                ...
```

Copyright 2019 MYOB Technology Pty Ltd

```
                    try
                    {
                        //Retrieving the list of customers with contacts
                        InitialDataRetrieval.ExportCustomers(screen);
                    }
                    ...
                }
            }
        }
    }
```

6 Rebuild the project and run the application. If you run the project in Debug mode, you can find the `Customers.csv` file with the results of the export in the `\bin\Debug` folder of the *MyBIIntegrationSBAPI* project.

You can open the project folder in File Explorer by right-clicking the project in Solution Explorer and selecting **Open Folder in File Explorer**.

A fragment of the resulting file is shown in the following screenshot.

	A	B	C	D	E	F	G	H	I	J
1	CustomerID	CustomerName	CustomerClass	Email	Phone1	AddressLine1	AddressLine2	City	State	PostalCode
2	C000000001	Jersey Central Office Equip	DEFAULT	jersey-equip@mail.con	+1 (777) 283-0414	1 De Villiers & Harrison St, 11-th Flr.		Johannesburg		
3	C000000002	Microchip Restaurant	DEFAULT		+1 (777) 459-4255	1 Kalisa Way		Paramus	NJ	
4	C000000003	Jevy Computers	DEFAULT	info@jevy-comp.con	+1 (777) 380-0089	1000 Pennsylvania Ave		San Francisco	CA	94107-3479
5	C000000004	KRK Consulting Service	DEFAULT	consulting@krk.con	+1 (777) 829-6988	1 Penn Plz		New York	NY	10119
6	C000000005	Wright Corner	DEFAULT	mail@wright-corner.con	+1 (777) 513-9166	1 W 89TH St		New York	NY	10024
7	C000000006	NETCAFE NY	DEFAULT	ny@netvafe.con	+1 (777) 673-7392	1011 High Ridge Rd		Stamford	CT	
8	C000000007	Bestype Image	DEFAULT		+1 (777) 966-6886	105 Cecil St		Singapore		

Figure: The `Customers.csv` file

Related Links

Screen-Based API Wrapper
API Objects Related to MYOB Advanced ERP Forms
Commands for Retrieving the Values of Elements
Export() Method

Copyright 2019 MYOB Technology Pty Ltd

Additional Information: Retrieval of Custom Fields

In a customization project, you can add custom fields to MYOB Advanced ERP forms. You can retrieve the values of these fields by using any of the integration interfaces described in this course. (This scenario is outside of the scope of this course but may be useful to some readers.)

Retrieval of Custom Fields Through OData

To retrieve custom fields through OData, you need to add these custom fields to the results of a generic inquiry exposed via OData.

Retrieval of Custom Fields Through the Contract-Based REST API

To retrieve custom fields through the contract-based REST API, you can use the `$custom` parameter. You can use the same approach to retrieve the values of any fields that are not included in the entity definition. For details about custom fields, see *Custom Fields* in the documentation. For details about the `$custom` parameter, see *Parameters for Retrieving Records*.

You can also add the custom fields to a custom endpoint or endpoint extension and work with them by using the same approach as was described in this lesson for the fields of the system endpoint. For details about custom endpoints and endpoint extensions, see *Custom Endpoints and Endpoint Extensions* in the documentation.

Retrieval of Custom Fields Through the Contract-Based SOAP API

To retrieve custom fields through the contract-based SOAP API, you should use the `CustomFields` property of an entity of the endpoint. You can use the same approach to retrieve the values of any fields that are not included in the entity definition. For details about custom fields, see *Custom Fields* in the documentation. For details about the `CustomFields` property, see *CustomFields Property*.

You can also add the custom fields to a custom endpoint or endpoint extension and work with them by using the same approach as was described in this lesson for the fields of the system endpoint. For details about custom endpoints and endpoint extensions, see *Custom Endpoints and Endpoint Extensions* in the documentation.

Retrieval of Custom Fields Through the Screen-Based SOAP API

To retrieve custom fields through the screen-based SOAP API, you need to generate a WSDL description of the service for the MYOB Advanced ERP form that contains custom fields after the publication of the customization project that has added these fields. After you have added the service reference to the project of your application, you can work with the custom fields in the same way as you work with the fields available on the MYOB Advanced ERP forms by default.

Copyright 2019 MYOB Technology Pty Ltd

Additional Information: Viewing of the Exposed Generic Inquiry Through an OData Client

The information in this topic applies only to OData.

You can view the data exposed through the OData interface by using external OData clients, such as Microsoft Excel and Microsoft Power BI.

For details about viewing the data with Microsoft Excel, see *To View Exposed Generic Inquiries in Excel*.

For details about viewing the data with Microsoft Power BI, see *Connecting to MYOB Advanced ERP from Power BI*.

Copyright 2019 MYOB Technology Pty Ltd

Lesson Summary

In this lesson, you have learned how to retrieve the data that is exposed in MYOB Advanced ERP on a data entry form by using the different integration interfaces.

The following table summarizes the options of data retrieval from a data entry form for each of the integration interfaces.

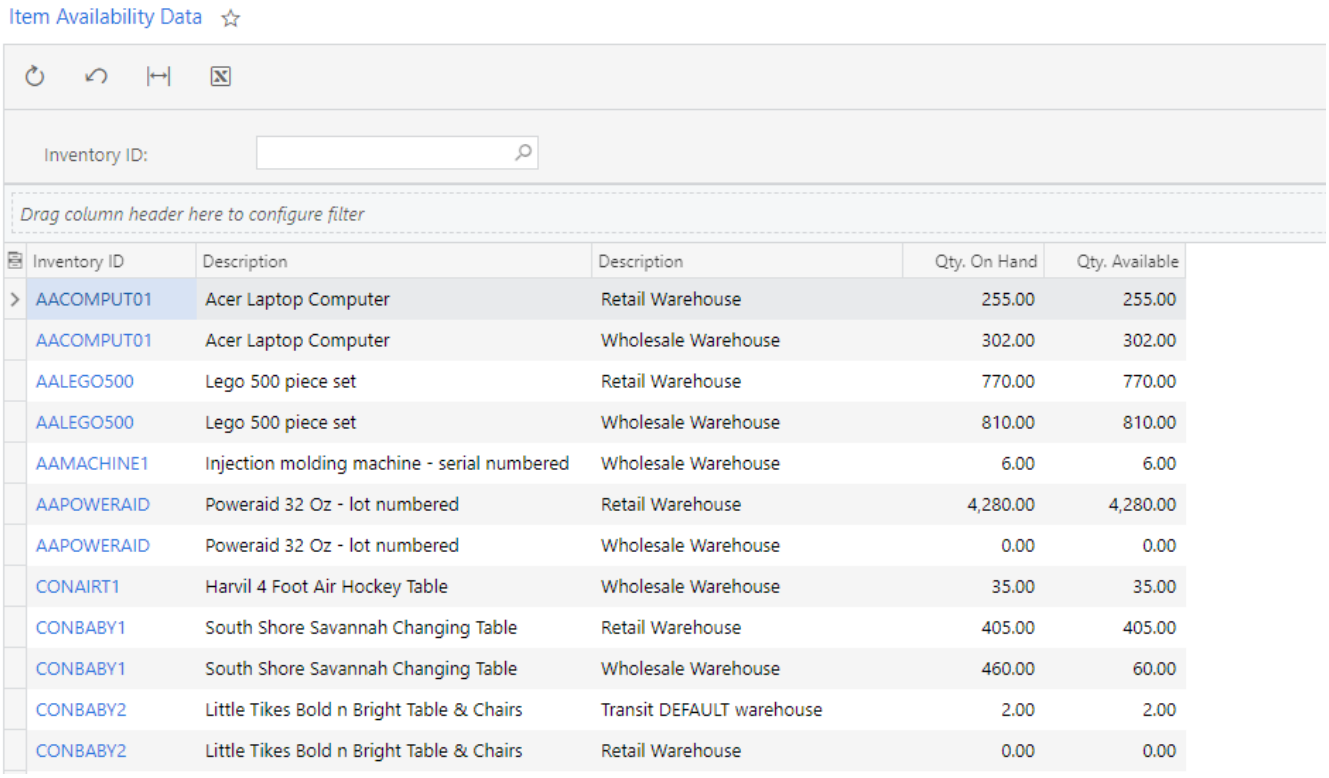
Integration Interface	Ability to Retrieve Data from a Data Entry Form	Ability to Retrieve Custom Fields
OData interface	No. You need to create a custom generic inquiry that retrieves the needed data.	Yes
REST API	Yes, if the form is mapped in an endpoint.	Yes
Contract-based SOAP API	Yes, if the form is mapped in an endpoint.	Yes
Screen-based SOAP API	Yes.	Yes

Copyright 2019 MYOB Technology Pty Ltd

Lesson 2.2: Retrieving the Quantities of Stock Items

The BI application of the MyStore company should display to a user the statistics of the items available in the warehouse. In MYOB Advanced ERP, you can view the item availability data on the Inventory Summary (IN401000) form. To view the on-hand and available quantities of stock items on this form, you select each item one by one on the form. You have to do the same if you retrieve the quantities of items through the web services API. (You cannot retrieve data from the Inventory Summary form through the OData protocol.) However, this way of data export has a drawback: It performs many requests to the MYOB Advanced ERP database; therefore, it has slow system performance.

In this lesson, you will use another way of exporting data from MYOB Advanced ERP. You will add to the MyBIIntegration application the method that exports data from the Item Availability Data generic inquiry, which has been preconfigured for this training course. This generic inquiry has been created based on the `PX.Objects.IN.INSiteStatus` database table and has one parameter, which you can use to filter the list of stock items by inventory ID. You can view this generic inquiry on the Generic Inquiry (SM208000) form by selecting the inquiry with the title *Item Availability Data* and clicking **View Inquiry**. The Item Availability Data generic inquiry is shown in the following screenshot.



Inventory ID	Description	Description	Qty. On Hand	Qty. Available
AACOMPUT01	Acer Laptop Computer	Retail Warehouse	255.00	255.00
AACOMPUT01	Acer Laptop Computer	Wholesale Warehouse	302.00	302.00
AALEGO500	Lego 500 piece set	Retail Warehouse	770.00	770.00
AALEGO500	Lego 500 piece set	Wholesale Warehouse	810.00	810.00
AAMACHINE1	Injection molding machine - serial numbered	Wholesale Warehouse	6.00	6.00
AAPOWERAID	Poweraid 32 Oz - lot numbered	Retail Warehouse	4,280.00	4,280.00
AAPOWERAID	Poweraid 32 Oz - lot numbered	Wholesale Warehouse	0.00	0.00
CONAIRT1	Harvil 4 Foot Air Hockey Table	Wholesale Warehouse	35.00	35.00
CONBABY1	South Shore Savannah Changing Table	Retail Warehouse	405.00	405.00
CONBABY1	South Shore Savannah Changing Table	Wholesale Warehouse	460.00	60.00
CONBABY2	Little Tikes Bold n Bright Table & Chairs	Transit DEFAULT warehouse	2.00	2.00
CONBABY2	Little Tikes Bold n Bright Table & Chairs	Retail Warehouse	0.00	0.00

Figure: Item Availability Data generic inquiry

Lesson Objective

In this lesson, you will learn how to retrieve the list of records from a generic inquiry form.

Copyright 2019 MYOB Technology Pty Ltd

Example 2.2.1: Using a Generic Inquiry (OData)

In this example, through OData, you will export the quantities of stock items from the Item Availability Data custom generic inquiry in MYOB Advanced ERP.

In the training data, the **Expose via OData** check box is selected for this generic inquiry in the Summary area of the Generic Inquiry (SM208000) form. This means that the generic inquiry is available through the OData interface.

In this example, you need to retrieve the full list of records of the generic inquiry. Although this generic inquiry has a parameter (which is optional), you cannot specify the parameter value of the inquiry in the OData request. The OData request returns the records for all possible variants of the parameters.

You will configure the request to return the data in JSON format by using the `$format` parameter.

Prerequisites

Before you send the request in the example, you need to sign in to MYOB Advanced ERP as described in *Example 1.2.1: Using OData*. If you have created a Postman collection with basic authentication configured, add the request described below to the collection and configure it to inherit the authorization type from the parent collection.

Instructions for Retrieving the Quantities of Items

To retrieve the quantities of items, do the following:

1. Configure the following settings of the request:
   - HTTP method: `GET`
   - URL: *http://localhost/MyStoreInstance/OData/Item%20Availability%20Data*
   - Parameter: `$format=json`
2. Send the request. If the request is successful, the response contains the `200 OK` status code. A fragment of the result is shown in the following code.

```
{
  "odata.metadata":
    "http://localhost/MyStoreInstance/odata/$metadata#Item%20Availability
%20Data",
  "value":
  [
    {
      "InventoryID": "SIMCARD    ",
      "Warehouse": "YOGI      ",
      "Description": "SIM card & contract",
      "QtyOnHand": "1000.000000",
      "QtyAvailable": "1000.000000"
    },
    {
      "InventoryID": "AACOMPUT01",
      "Warehouse": "MAIN      ",
      "Description": "Acer Laptop Computer",
      "QtyOnHand": "150.000000",
      "QtyAvailable": "150.000000"
    },
    ...
  ]
}
```

3. Save the request.

Related Links

OData Support

Copyright 2019 MYOB Technology Pty Ltd

Example 2.2.2: Using PUT to Retrieve Data from an Inquiry (REST)

In this example, through the REST API, you will configure an HTTP request that exports the quantities of stock items from the Item Availability Data custom generic inquiry in MYOB Advanced ERP.

Because no entity is mapped to the custom generic inquiry in the system endpoints, which are provided in MYOB Advanced ERP by default, to export data from this generic inquiry by using the contract-based API, you need to use a custom endpoint or an extension of the existing endpoint. You will use the *ItemAvailabilityData/0001* custom endpoint. (The creation of custom endpoints and endpoint extensions is outside of the scope of this course. For more details, see *Additional Information: Custom Endpoints and Endpoint Extensions*.)

You will use the `PUT` HTTP method and the `ItemAvailabilityDataInquiry` entity of the custom *ItemAvailabilityData/0001* endpoint. The `ItemAvailabilityDataInquiry` entity is mapped to the Item Availability Data custom generic inquiry.

You will use the `$expand` parameter to specify the nested detail entity (`Results`) to be returned. To retrieve the data from a generic inquiry you have to expand the nested detail entity with the results of the generic inquiry. This entity is usually called `Results`.

Although this generic inquiry has a parameter (which is optional), you will not specify the parameter value of the inquiry in this example, because you need to retrieve the full list of records of the generic inquiry. Therefore, you will pass no values in the body of the request to specify the parameter.

Prerequisites

Before you can test the example, you need to sign in to MYOB Advanced ERP as described in *Example 1.2.2: Using the REST API*. For the request in this example, you have to pass the cookies that you have received during the sign-in.

Instructions for Retrieving the Quantities of Items

To retrieve the quantities of items, do the following:

1. In Postman, configure the following settings of the contract-based REST API request:
   - HTTP method: `PUT`
   - URL: *http://localhost/MyStoreInstance/entity/ItemAvailabilityData/0001/ItemAvailabilityDataInquiry*
   - Parameter: `$expand=Results`
   - Headers:

Key	Value
`Accept`	`application/json`
`Content-Type`	`application/json`

   - Body:

```
{

}
```

Copyright 2019 MYOB Technology Pty Ltd

2. Send the request. If the request is successful, the response contains the `200 OK` status code and includes the result of generic inquiry in JSON format. The following code example shows a fragment of the response body.

```
{
    "id": "51c779ee-a54a-4623-9cac-64ee87b4589e",
    "rowNumber": 1,
    "note": null,
    "InventoryID": {},
    "Results": [
        {
            "id": "88e44fc9-96f8-43b4-b93d-225ef1e614ed",
            "rowNumber": 1,
            "note": null,
            "Description": {
                "value": "SIM card & contract"
            },
            "InventoryID": {
                "value": "SIMCARD"
            },
            "QtyAvailable": {
                "value": 1000
            },
            "QtyOnHand": {
                "value": 1000
            },
            "Selected": {
                "value": false
            },
            "Warehouse": {
                "value": "YOGI"
            },
            "custom": {},
            "files": []
        },
        ...
    ]
}
```

Related Links

Retrieval of Data from an Inquiry Form
Parameters for Retrieving Records

Copyright 2019 MYOB Technology Pty Ltd

Example 2.2.3: Using the Put() Method to Retrieve Data from an Inquiry (Contract-Based SOAP)

In this example, through the contract-based SOAP API, you will export the quantities of stock items from the Item Availability Data custom generic inquiry in MYOB Advanced ERP.

Because no entity is mapped to the custom generic inquiry in the system endpoints, which are provided in MYOB Advanced ERP by default, to export data from this generic inquiry by using the contract-based API, you need to use a custom endpoint or an extension of the existing endpoint. You will use the *ItemAvailabilityData/0001* custom endpoint, which has been preconfigured for this course.

The creation of custom endpoints and endpoint extensions is outside of the scope of this course. For more details, see *Additional Information: Custom Endpoints and Endpoint Extensions*.

You will add the *ItemAvailabilityData* service reference and use the `ItemAvailabilityDataInquiry` class and the `Put()` method of an instance of the `DefaultSoapClient` class available in this service reference. The `ItemAvailabilityDataInquiry` class corresponds to the `ItemAvailabilityDataInquiry` entity of the *ItemAvailabilityData/0001* endpoint. This entity is mapped to the Item Availability Data custom generic inquiry.

You will set the `ReturnBehavior` property of the `Results` detail entity to `ReturnBehavior.All` to retrieve the values of all columns of the generic inquiry.

Although this generic inquiry has a parameter (which is optional), you will not specify the parameter value of the inquiry in this example because you need to retrieve the full list of records of the generic inquiry.

Prerequisites

In the *MyBIIntegration* project, add the *ItemAvailabilityData* service reference for the *ItemAvailabilityData/0001* custom endpoint, and configure the application to use cookies for this service reference. For details about how to add the reference and configure the application, see *Example 1.1.1. Configuring the Client Application (Contract-Based SOAP)*.

Instructions for Retrieving the Quantities of Items

To retrieve the quantities of items, do the following:

1. In the `Integration` folder of the *MyBIIntegration* project, add a new C# class with the name `InitialDataRetrievalGI`.
2. In the `InitialDataRetrievalGI.cs` file, type the `using` directives as shown in the following code to make the `InitialDataRetrievalGI` class use the *ItemAvailabilityData* service reference and the `System.IO` classes.

```
using MyBIIntegration.ItemAvailabilityData;
using System.IO;
```

3. In the `InitialDataRetrievalGI` class, define the `RetrieveItemQuantities()` method, as the following code shows.

```
//Retrieving the quantities of stock items
public static void RetrieveItemQuantities(
    DefaultSoapClient soapClient)
{
    Console.WriteLine("Retrieving the quantities of stock items...");

    //Return all columns of the generic inquiry
    ItemAvailabilityDataInquiry inqParameters =
      new ItemAvailabilityDataInquiry
    {
        Result = new Results[]
        {
```

Copyright 2019 MYOB Technology Pty Ltd

```
            new Results
            {
                ReturnBehavior = ReturnBehavior.All
            }
        }
    };

    //Retrieve the quantities of items
    ItemAvailabilityDataInquiry results =
        (ItemAvailabilityDataInquiry)soapClient.Put(inqParameters);

    //Save the results to a CSV file
    using (StreamWriter file = new StreamWriter("ItemAvailabilityData.csv"))
    {
        //Add headers to the file
        file.WriteLine(
            "InventoryID;Description;Warehouse;QtyAvailable;QtyOnHand;");

        //Write the values for each item
        foreach (Results result in results.Result)
        {
            file.WriteLine(
                string.Format("{0};{1};{2};{3};{4};",
                result.InventoryID.Value,
                result.Description.Value,
                result.Warehouse.Value,
                result.QtyAvailable.Value,
                result.QtyOnHand.Value));
        }
    }
}
```

4 In the `Program.cs` file, add the `using` directive and the code that calls the `RetrieveItemQuantities()` method, as shown in the following code. Notice that you use an instance of the `DefaultSoapClient` class from the *ItemAvailabilityData* service reference.

You can comment the code of the previous example in the `Program.Main()` method.

```
...
using MyBIIntegration.ItemAvailabilityData;

namespace MyBIIntegration
{
  class Program
  {
    static void Main(string[] args)
    {
      //Using the Default/18.200.001  endpoint
      using (Default.DefaultSoapClient soapClient =
          new Default.DefaultSoapClient())
      {
        ...
      }

      //Using the ItemAvailabilityData/0001 endpoint
      using (ItemAvailabilityData.DefaultSoapClient soapClient =
        new ItemAvailabilityData.DefaultSoapClient())
      {
        //Sign in to MYOB Advanced ERP
        soapClient.Login
        (
          Properties.Settings.Default.Username,
          Properties.Settings.Default.Password,
          null, null, null
        );
```

Copyright 2019 MYOB Technology Pty Ltd

```
        try
        {
          //Retrieving the quantities of stock items
          InitialDataRetrievalGI.RetrieveItemQuantities(soapClient);
        }
        catch (Exception e)
        {
          Console.WriteLine(e);
          Console.WriteLine();
          Console.WriteLine("Press any key to continue");
          Console.ReadLine();
        }
        finally
        {
          //Sign out from MYOB Advanced ERP
          soapClient.Logout();
        }
      }
    }
  }
}
```

5. Rebuild the project, and run the application. If you run the project in Debug mode, you can find the `ItemAvailabilityData.csv` file with the results of the export in the `\bin\Debug` folder of the MyBIIntegration project. The resulting file is shown in the following screenshot.

	A	B	C	D	E
1	InventoryID	Description	Warehouse	QtyAvailable	QtyOnHand
2	SIMCARD	SIM card & contract	YOGI	1000	1000
3	AACOMPUT01	Acer Laptop Computer	MAIN	150	150
4	AALEGO500	Lego, 500 piece set	MAIN	1974	2000
5	AAMACHINE1	Injection molding machine	MAIN	127	130
6	CONGRILL	Char-Broil Classic 480	MAIN	193	200
7	CONTABLE1	Folding Picnic Table 6 Foot	MAIN	195	200
8	LAPTOP12	Laptop 12"	MAIN	20	20
9	LAPTOP14	Laptop 14"	MAIN	20	20
10	MOUSE	Computer mouse	MAIN	40	40
11	HEADSET	Headset	MAIN	40	40
12	KEYBOARD	Computer keyboard	MAIN	40	40
13	MONITOR26	Monitor 26"	MAIN	40	40
14	LEATHCASE	Leather case for laptops 12"-15	MAIN	40	40

Figure: The `ItemAvailabilityData.csv` file

Related Links

Put() Method

ReturnBehavior Property (Contract Version 3)

Copyright 2019 MYOB Technology Pty Ltd

Example 2.2.4: Using the Export() Method to Retrieve Data from an Inquiry (Screen-Based SOAP)

In this example, through the screen-based SOAP API, you will export the quantities of stock items from the Item Availability Data (INGI0002) custom generic inquiry in MYOB Advanced ERP.

The Item Availability Data generic inquiry has the *INGI0002* form ID. Therefore, to access this form, you will use the classes and methods with the *INGI0002* prefix.

To specify the fields to be retrieved, you will configure an array of `Command` objects. In this array, you will specify the fields by using the schema of the Item Availability Data generic inquiry available in the `Content` object. The results of the generic inquiry are available in the `Result` subobject of the `Content` object.

You will use the `Export()` method of the `Screen` object to retrieve the values of the specified fields. You will specify the parameters of the method as follows:

- Pass the configured array of commands as the first parameter.
- Because you need to retrieve the full list of items, pass `null` as the second parameter (you do not need to apply any filter to the list), and pass `0` as the third parameter (you do not need to limit the number of records).
- Pass `true` as the fourth parameter, to include column headers in the result of export. The first row of the exported data will include the names of exported elements in the first row.
- Pass `false` as the fifth parameter, because you do not need to stop the export if an error occurs during this process.

Retrieving the List of Stock Items with Quantities

To export item availability data, proceed as follows:

1. Add the `InitialDataRetrievalGI` class to the `Integration` folder of the *MyBIIntegrationSBAPI* project.
2. In the `InitialDataRetrievalGI.cs` file, type the `using` directives as shown in the following code to make the `InitialDataRetrievalGI` class use the *MyBIIntegration* web reference and the classes of the `System.IO` namespace.

```
using MyBIIntegrationSBAPI.MyBIIntegration;
using System.IO;
```

3. In the `InitialDataRetrievalGI` class, define the `ExportItemQuantities()` method, as the following code shows.

```
//Retrieving the item availability data by using a generic inquiry
public static void ExportItemQuantities(Screen screen)
{
    Console.WriteLine("Retrieving stock item quantities...");

    //Get the schema of the Item Availability Data generic inquiry
    //by using the screen-based API wrapper
    INGI0002Content schema =
      PX.Soap.Helper.GetSchema<INGI0002Content>(screen);

    //Configure the sequence of commands
    var commands = new Command[]
    {
        schema.Result.Warehouse,
        schema.Result.InventoryID,
        schema.Result.Description,
        schema.Result.QtyOnHand,
        schema.Result.QtyAvailable,
    };
```

Copyright 2019 MYOB Technology Pty Ltd

```
    //Retrieve the list of items
    String[][] items = screen.INGI0002Export(commands, null, 0, true, false);

    //Save the data to a CSV file
    StreamWriter file = new StreamWriter("ItemAvailabilityData.csv");
    {
        foreach (string[] rows in items)
        {
            foreach (string row in rows)
            {
                file.Write(row + ";");
            }
            file.WriteLine();
        }
    }
    file.Close();
}
```

4 In the `try` block of the `Main()` method of the `Program` class, call the `ExportItemQuantities()` method of the `InitialDataRetrievalGI` class, as the following code shows.

```
InitialDataRetrievalGI.ExportItemQuantities(screen);
```

5 Rebuild the project, and run the application. You can find the `ItemAvailabilityData.csv` file with the results of the export in the `\bin\Debug` folder of the *MyBIIntegrationSBAPI* project if you run the project in Debug mode.

The resulting file is shown in the following screenshot.

	A	B	C	D	E
1	Warehouse	InventoryID	Description	QtyOnHand	QtyAvailable
2	YOGI	SIMCARD	SIM card & contract	1000.000000	1000.000000
3	MAIN	AACOMPUT01	Acer Laptop Computer	150.000000	150.000000
4	MAIN	AALEGO500	Lego, 500 piece set	2000.000000	1974.000000
5	MAIN	AAMACHINE1	Injection molding machine	130.000000	127.000000
6	MAIN	CONGRILL	Char-Broil Classic 480	200.000000	193.000000
7	MAIN	CONTABLE1	Folding Picnic Table 6 Foot	200.000000	195.000000
8	MAIN	LAPTOP12	Laptop 12"	20.000000	20.000000
9	MAIN	LAPTOP14	Laptop 14"	20.000000	20.000000
10	MAIN	MOUSE	Computer mouse	40.000000	40.000000
11	MAIN	HEADSET	Headset	40.000000	40.000000
12	MAIN	KEYBOARD	Computer keyboard	40.000000	40.000000
13	MAIN	MONITOR26	Monitor 26"	40.000000	40.000000
14	MAIN	LEATHCASE	Leather case for laptops 12"-15"	40.000000	40.000000

Figure: `ItemAvailabilityData.csv` file

Related Links

Screen-Based API Wrapper
API Objects Related to MYOB Advanced ERP Forms
Commands for Retrieving the Values of Elements
Export() Method

Copyright 2019 MYOB Technology Pty Ltd

Lesson Summary

In this lesson, you have learned how to retrieve the list of records that is exposed in MYOB Advanced ERP on a generic inquiry form by using different integration interfaces.

The following table summarizes the options of data retrieval from a generic inquiry form for each of the integration interfaces.

Integration Interface	Ability to Retrieve Data from a Generic Inquiry Without Parameters	Ability to Specify the Values of Parameters of Generic Inquiries
OData interface	Yes	No
REST API	Yes, if the generic inquiry is mapped in an endpoint	Yes, if the generic inquiry and its parameters are mapped in an endpoint
Contract-based SOAP API	Yes, if the generic inquiry is mapped in an endpoint	Yes, if the generic inquiry and its parameters are mapped in an endpoint
Screen-based SOAP API	Yes	Yes

Copyright 2019 MYOB Technology Pty Ltd

Part 3: Retrieval of the Delta of Records

The BI application of the MyStore company displays information about the stock items registered in MYOB Advanced ERP. To provide actual data about the stock items to the BI system, the MyBIIntegration application needs to retrieve this information once a day. To optimize the performance of the MyBIIntegration application, the MyBIIntegration application retrieves only the data about the items that have been modified within the past day.

The BI application also needs to display information about item availability in MYOB Advanced ERP in real time. To display actual data about the stock item availability, the MyBIIntegration application uses push notifications configured in MYOB Advanced ERP.

By completing the examples of this part, you will configure the MyBIIntegration application to retrieve only the modified records from MYOB Advanced ERP. You can use all available integration interfaces for the export. You will also configure push notifications to be sent by MYOB Advanced ERP when item availability data has been changed.

Copyright 2019 MYOB Technology Pty Ltd

Lesson 3.1: Retrieving the List of Modified Stock Items

The BI application of the MyStore company should display to each user the information about the items that are sold in the store. These items are entered and updated on the Stock Items (IN202500) form in MYOB Advanced ERP. The MyBIIntegration application already has the list of items with the necessary information and needs to retrieve only the changes to the items that have been made during the past
day. To display the list of modified items to a user, the MyBIIntegration application should export the list of stock items that satisfy the specified condifions from MYOB Advanced ERP. You will export the stock item records that have the *Active* status code and that were modified within the past day.

For a stock item record entered and maintained on the Stock Items form, you will export the following values:

- The inventory ID (the **Inventory ID** box in the Summary area)
- The description of the item (**Description** of the Summary area)
- The item class assigned to the item in MYOB Advanced ERP (the **Item Class** box of the **Item Defaults** section of the **General Settings** tab)
- The base unit of measurement (the **Base Unit** box of the **Unit of Measure** section of the **General Settings** tab)
- The date and time the record was last modified (no corresponding elements on the form)
- Information about the availability of the item in particular warehouses (the **Warehouse Details** tab):
    - The warehouse ID (the **Warehouse** column)
    - The quantity of the item available in the warehouse (the **Qty. On Hand** column)

The elements that are available on the form are shown in the following screenshot. The value of the last modified date and time will be exported from an internal field, which is not available among the elements of the form.

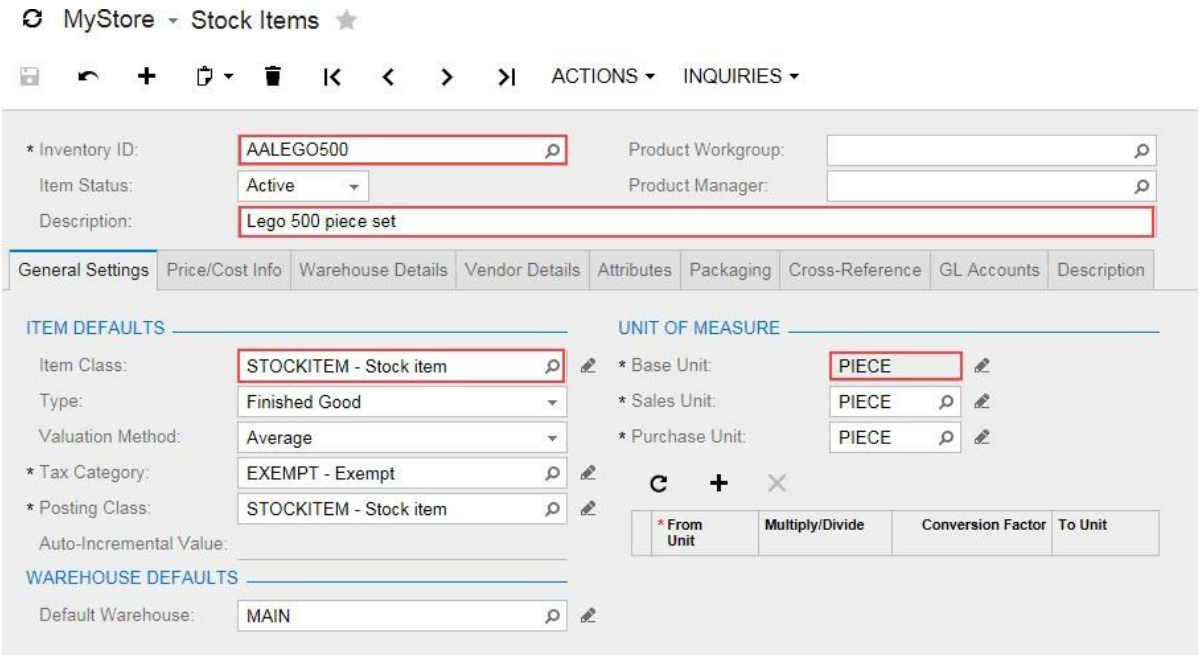


Figure: Elements whose values will be exported

Copyright 2019 MYOB Technology Pty Ltd

This lesson shows how you can implement this scenario by using the OData interface, contract-based REST API, contract-based SOAP API, and screen-based SOAP API.

Lesson Objective

In this lesson, you will learn how to retrieve the list of records that were modified within a period.

Copyright 2019 MYOB Technology Pty Ltd

Prerequisites

To have at least one stock item record modified within the past day, you need to perform the actions described in this topic before you perform the examples of this lesson.

On the Stock Items (IN202500) form, modify the *AACOMPUT01* and *AALEGO500* inventory items as follows:

- Change the status of the *AACOMPUT01* inventory item to *Inactive* and save the record.
- Change the description of the *AALEGO500* inventory item to `Lego, 500 piece set`, and save the record.

Now you have at least two inventory items that were modified within the past month, and one of them has *Active* status.

The system tracks the last modified date for every record, but this date is not displayed on the Stock Items form. In the system, there is a preconfigured generic inquiry where you can check the dates when stock items were last modified. To view this generic inquiry, on the Generic Inquiry (SM208000) form, select the inquiry with the title *Stock Items: Last Modified Date*, and click **View Inquiry** on the form toolbar. You can sort items by the last modified date on the Last Modified Date (INGI0001) generic inquiry, as shown in the following screenshot.

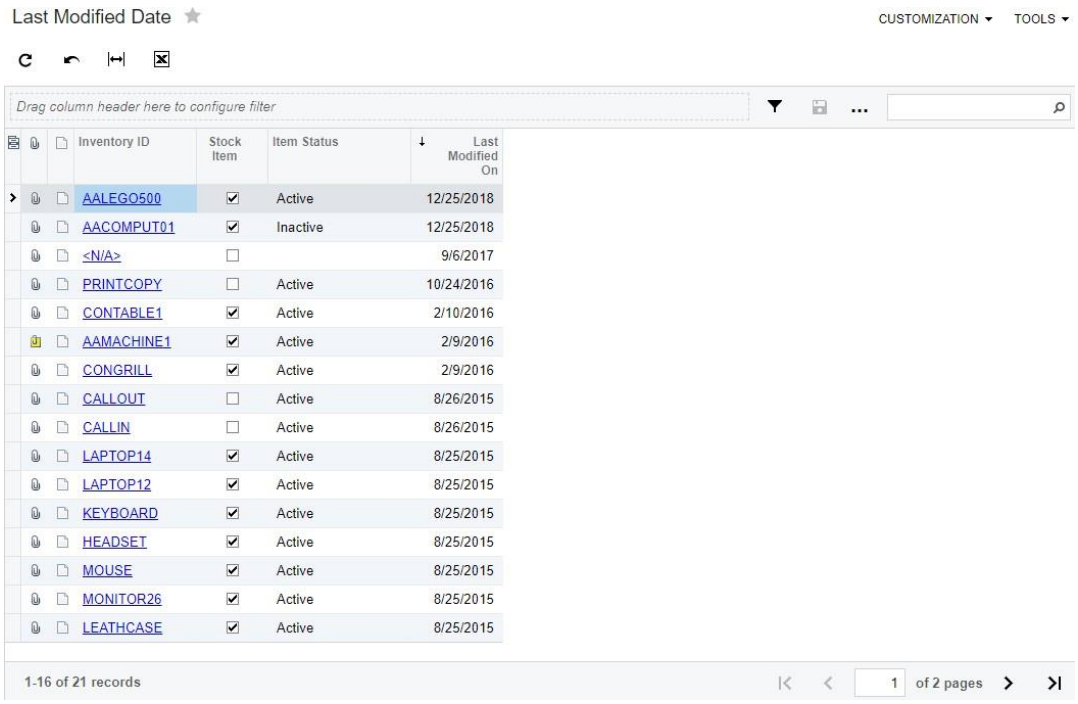


Figure: Last Modified Date generic inquiry

Copyright 2019 MYOB Technology Pty Ltd

Example 3.1.1: Filtering the Result of a Generic Inquiry (OData)

In this example, through the OData protocol, you will export the list of active stock item records that were modified within the past day.

Through the OData protocol, you cannot export records directly from a data entry form, such as the Stock Items (IN202500) form. Thus, before the export, you have to configure a generic inquiry that retrieves the needed data from MYOB Advanced ERP.

In this example, you will use the Modified Stock Items custom generic inquiry. This generic inquiry has no parameters and is based on the `PX.Objects.IN.INSiteStatus` and `PX.Objects.IN.InventoryItem` database tables. This generic inquiry contains all the fields that should be retrieved. (Creation of generic inquiries is outside of the scope of this course.)

In the training data, the **Expose via OData** check box is selected for this generic inquiry in the Summary area of the Generic Inquiry form. This means that the generic inquiry is available through the OData interface.

Because you need to filter the results of the inquiry to obtain only the active records that were modified within the past day, you need to use the `$filter` parameter. Before you configure the filter, you will find out the names of the fields that are available in the generic inquiry by using a specific request to the OData interface.

You will use the `$filter` parameter to specify the search conditions for the fields. You use *OData URI conventions* to specify the conditions.

You will obtain the data in JSON format by using the `$format` parameter of the request.

Prerequisites

Before you send the request in the example, you need to sign in to MYOB Advanced ERP as described in *Example 1.2.1: Using OData*. If you have created a Postman collection with basic authentication configured, add the request described below to the collection and configure it to inherit the authorization type from the parent collection.

Instructions for Retrieving the Names of the Fields of the Generic Inquiry

To obtain the list of fields of the generic inquiry, do the following:

1. Configure the following settings of the request:
    - HTTP method: `GET`
    - URL: *http://localhost/MyStoreInstance/OData/$metadata*
2. Send the request. If the request is successful, its response contains the `200 OK` status code. In the response body, find the `EntityType` tag with `Name="Modified Stock Items"`, and find the properties with the names `ItemStatus` and `LastModifiedOn`, which are shown in the following screenshot.

Copyright 2019 MYOB Technology Pty Ltd

```
154             <Property Name="QtyOnHand" Type="Edm.Decimal" />
155             <Property Name="QtyAvailable" Type="Edm.Decimal" />
156         </EntityType>
157         <EntityType Name="Modified Stock Items">
158             <Key>
159                 <PropertyRef Name="InventoryID" />
160                 <PropertyRef Name="Warehouse" />
161             </Key>
162             <Property Name="InventoryID" Type="Edm.String" />
163             <Property Name="Warehouse" Type="Edm.String" />
164             <Property Name="Description" Type="Edm.String" />
165             <Property Name="ItemStatus" Type="Edm.String" />
166             <Property Name="LastModifiedOn" Type="Edm.DateTime" />
167             <Property Name="ItemClass" Type="Edm.String" />
168             <Property Name="BaseUnit" Type="Edm.String" />
169             <Property Name="QtyOnHand" Type="Edm.Decimal" />
170         </EntityType>
```

Figure: Fields of the generic inquiry

You will use these field names to configure a filter for the results of the generic inquiry through the OData protocol.

3. Save the request.

Instructions for Retrieving the List of Modified Stock Items

To retrieve the list of modified stock items, do the following:

1. Configure the following settings of the request:
   - HTTP method: `GET`
   - URL: *http://localhost/MyStoreInstance/OData/Modified%20Stock%20Items*
   - Parameters:

Parameter	Value
`$filter`	`ItemStatus eq 'Active' and LastModifiedOn gt datetime'2018-11-31T00:00:00.000'` Specify today's date instead of `2018-11-31`.
`$format`	`json`

2. Send the request. The response of the successful request contains the `200 OK` status code. The following code shows an example of the response body.

```
{
  "odata.metadata":
    "http://localhost/MyStoreInstance/odata/$metadata#Modified%20Stock
%20Items",
  "value":
  [
    {
      "InventoryID": "AALEGO500 ",
      "Warehouse": "MAIN      ",
      "Description": "Lego, 500 piece set",
      "ItemStatus": "Active",
      "LastModifiedOn": "2018-11-12T14:38:43.557",
      "ItemClass": "STOCKITEM ",
      "BaseUnit": "PIECE",
      "QtyOnHand": "2000.000000"
    }
  ]
}
```

3. Save the request.

Copyright 2019 MYOB Technology Pty Ltd

Related Links

OData Support

Copyright 2019 MYOB Technology Pty Ltd

Example 3.1.2: Using GET and the LastModified Field (REST)

In this example, through the REST API, you will configure an HTTP request that exports the list of active stock item records that were modified within the past day.

You will use the `GET` HTTP method and the `StockItem` entity of the *Default/18.200.001* endpoint to list the stock items. The `StockItem` entity is mapped to the Stock Items (IN202500) form.

You will use the `$filter` parameter to specify the search conditions for the `Active` and `LastModified` fields. You use *OData URI conventions* to specify the conditions.

Because the database can contain thousands of stock item records, to achieve the best performance of the application, you need to specify the fields of the stock item records that should be returned. You will use the `$select` parameter to specify the fields whose values should be retrieved from MYOB Advanced ERP for each stock item record. For a stock item record, you will export the values of the following fields of the `StockItem` entity:

- `InventoryID`
- `Description`
- `ItemClass`
- `BaseUOM`
- `WarehouseDetails` (nested `StockItemWarehouseDetail` entity):
  - `WarehouseID`
  - `QtyOnHand`

The `LastModified` and `ItemStatus` fields will be returned because they are specified in the `$filter` parameter.

You will use the `$expand` parameter to specify the `WarehouseDetails` nested detail entity to be returned.

Prerequisites

Before you can test the example, you need to sign in to MYOB Advanced ERP as described in *Example 1.2.2: Using the REST API*. For the request in this example, you have to pass the cookies that you have received during the sign-in.

Instructions for Retrieving the List of Modified Stock Items

To retrieve the list of modified stock items, do the following:

1. In Postman, configure the following settings of the contract-based REST API request:
   - HTTP method: `GET`
   - URL: *http://localhost/MyStoreInstance/entity/Default/18.200.001/StockItem*
   - Parameters:

Parameter	Value
`$filter`	`ItemStatus eq 'Active' and LastModified gt datetimeoffset'2018-11-31T00:00:00.000'` Specify today's date instead of `2018-11-31`.
`$expand`	`WarehouseDetails`

Copyright 2019 MYOB Technology Pty Ltd

Parameter	Value
`$select`	`InventoryID,Description,WarehouseDetails/WarehouseID,WarehouseDetails/QtyOnHand,ItemClass,BaseUOM`

- Headers:

Key	Value
`Accept`	`application/json`
`Content-Type`	`application/json`

2. Send the request. If the request is successful, its response contains the `200 OK` status code and includes the list of requested fields of the active stock item records modified within the past day in JSON format. The following code example shows an example of the response body.


```
[
    {
        "id": "b9be8b17-64fe-4400-b91d-b7966f8030da",
        "rowNumber": 1,
        "note": "",
        "BaseUOM": {
            "value": "PIECE"
        },
        "Description": {
            "value": "Lego, 500 piece set"
        },
        "InventoryID": {
            "value": "AALEGO500"
        },
        "ItemClass": {
            "value": "STOCKITEM"
        },
        "ItemStatus": {
            "value": "Active"
        },
        "LastModified": {
            "value": "2018-11-31T13:55:48.097+03:00"
        },
        "WarehouseDetails": [
            {
                "id": "ba44dff3-8486-4b33-88a3-55096acb9988",
                "rowNumber": 1,
                "note": "",
                "QtyOnHand": {
                    "value": 2000
                },
                "WarehouseID": {
                    "value": "MAIN"
                },
                "custom": {},
                "files": []
            }
        ],
        "custom": {},
        "files": []
    }
]
```


3. Save the request.

Related Links

Retrieval of Records by Conditions
Parameters for Retrieving Records

Copyright 2019 MYOB Technology Pty Ltd

Example 3.1.3: Using GetList and the LastModified Field (Contract-Based SOAP)

In this example, you will add to the MyBIIntegration application a method that exports stock items from the Stock Items (IN202500) form. You will export the stock item records that have the *Active* status and that were modified within the past day.

You will use the `StockItem` class and the `GetList()` method of an instance of the `DefaultSoapClient` class available in the *Default* service reference for the *Default/18.200.001* endpoint. The `StockItem` class corresponds to the `StockItem` entity of the endpoint, which is mapped to the Stock Items form.

You will use the `StringSearch` and `DateTimeSearch` objects to specify the search conditions for the `Active` and `LastModified` fields, respectively.

Because the database can contain thousands of stock item records, to achieve the best performance of the application, you need to specify the fields of the stock item records that should be returned. You will specify the fields to be returned for each stock item record by using the `StringReturn` and `DecimalReturn` objects. For a stock item record, you will export the values of the following fields of the `StockItem` entity:

- `InventoryID`
- `Description`
- `ItemClass`
- `BaseUOM`
- `WarehouseDetails` (nested `StockItemWarehouseDetail` entity):
    - `WarehouseID`
    - `QtyOnHand`

The `LastModified` and `ItemStatus` fields will be returned because they are also specified in the request (with the `StringSearch` and `DateTimeSeach` objects).

You will use set the `ReturnBehavior` property to `ReturnBehavior.OnlySpecified` for the `StockItem` entity and the nested entity, to make the service return only the fields that you have specified in the request.

Retrieving the List of Stock Items

To retrieve the list of stock items, do the following:

1. In the `Integration` folder of the *MyBIIntegration* project, add a new C# class, `RetrievalOfDelta`.
2. In the `RetrievalOfDelta.cs` file, type the `using` directives as shown in the following code to make the `RetrievalOfDelta` class use the *Default* service reference and the `System.IO` classes.

```
using MyBIIntegration.Default;
using System.IO;
```

3. In the `RetrievalOfDelta` class, define the `ExportStockItems()` method, as the following code shows.

```
//Retrieving the list of stock items
public static void ExportStockItems(DefaultSoapClient soapClient)
{
    Console.WriteLine("Retrieving the list of stock items...");

    //Specify the parameters of stock items to be returned
    StockItem stockItemsToBeFound = new StockItem
```

Copyright 2019 MYOB Technology Pty Ltd

```
    {
        //Specify return behavior
        ReturnBehavior = ReturnBehavior.OnlySpecified,

        //Filter the items by the last modified date and status
        LastModified = new DateTimeSearch
        {
            Value = DateTime.Now.AddDays(-1),
            Condition = DateTimeCondition.IsGreaterThan
        },
        ItemStatus = new StringSearch { Value = "Active" },

        //Specify other fields to be returned
        InventoryID = new StringReturn(),
        Description = new StringReturn(),
        ItemClass = new StringReturn(),
        BaseUOM = new StringReturn(),
        WarehouseDetails = new StockItemWarehouseDetail[]
        {
            new StockItemWarehouseDetail
            {
                ReturnBehavior = ReturnBehavior.OnlySpecified,
                WarehouseID = new StringReturn(),
                QtyOnHand = new DecimalReturn()
            }
        }
    };

    //Get the list of stock items
    Entity[] stockItems = soapClient.GetList(stockItemsToBeFound);

    //Save the results to a CSV file
    using (StreamWriter file = new StreamWriter("StockItems.csv"))
    {
        //Add headers to the file
        file.WriteLine("InventoryID;Description;ItemClass;BaseUOM;" +
          "LastModified;WarehouseID;QtyOnHand;");

        //Write the values for each item
        foreach (StockItem stockItem in stockItems)
        {
            foreach (
              StockItemWarehouseDetail detail in stockItem.WarehouseDetails)
            {
                file.WriteLine(string.Format("{0};{1};{2};{3};{4};{5};{6}",
                    stockItem.InventoryID.Value,
                    stockItem.Description.Value,
                    stockItem.ItemClass.Value,
                    stockItem.BaseUOM.Value,
                    stockItem.LastModified.Value,
                    detail.WarehouseID.Value,
                    detail.QtyOnHand.Value));
            }
        }
    }
}
```

4 In the `Program.cs` file, in the `try` block that uses the *Default* service, call the `ExportStockItems()` method of the `RetrievalOfDelta` class, as the following code shows.

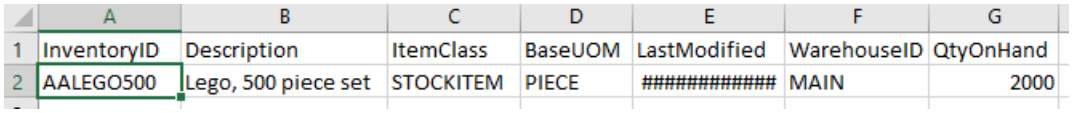
```
using (Default.DefaultSoapClient soapClient = new Default.DefaultSoapClient())
{
    ...

    try
    {
        //Retrieving the list of stock items modified within the past day
        RetrievalOfDelta.ExportStockItems(soapClient);
```

Copyright 2019 MYOB Technology Pty Ltd

```
        }
        ...
}
```

5. Rebuild the project, and run the application. If you run the project in Debug mode, you can find the `StockItems.csv` file with the results of the export in the `\bin\Debug` folder of the *MyBIIntegration* project. The resulting file is shown in the following screenshot.



	A	B	C	D	E	F	G
1	InventoryID	Description	ItemClass	BaseUOM	LastModified	WarehouseID	QtyOnHand
2	AALEGO500	Lego, 500 piece set	STOCKITEM	PIECE	############	MAIN	2000

Figure: `StockItems.csv` file

Related Links

GetList() Method (Contract Version 3)
ReturnBehavior Property (Contract Version 3)

Copyright 2019 MYOB Technology Pty Ltd

Example 3.1.4: Using Export and LastModifiedDateTime (Screen-Based SOAP)

In this example, you will add to the MyBIIntegration application a method that exports stock items from the Stock Items (IN202500) form. You will export the stock item records that have the *Active* status and that were modified within the past day.

The Stock Items form has the *IN202500* form ID. Therefore, to access this form, you will use the classes and methods with the *IN202500* prefix.

You will set the current thread culture by using the `SetLocaleName()` method of a `Screen` object. You should specify the current culture to make MYOB Advanced ERP correctly recognize the dates when the needed stock items were modified.

By default, all country-specific data in MYOB Advanced ERP uses the invariant culture, which is similar to the English (United States) culture. The `ToLongDateString()` system method, which is used in the filter code in this example, is culture-sensitive and returns the date in the current culture format. If current culture is not English (United States) and the current culture is not set by using the `SetLocaleName()` method, the date value can be interpreted incorrectly by MYOB Advanced ERP.

You will use the screen-based API wrapper to prevent application failures due to possible UI changes on the Stock Items form in later versions of MYOB Advanced ERP. With the `GetSchema()` static method of the wrapper, you will receive a `Content` object that corresponds to the Stock Items form.

To specify the fields to be retrieved, you will configure an array of `Command` objects. In the `commands` array, you will add the `EveryInventoryID` service command, which is available through `ServiceCommands` of `StockItemSummary`, to make MYOB Advanced ERP export all stock item records available on the form. In this array, you will specify the fields by using the schema of the Stock Items form available in the `Content` object. For the `LastModifiedDateTime` field, which is not available on the form, you will create a new `Field` object.

To define the filters for the stock item records, you will create an array of `Filter` objects and add the following filters to it:

- The filter that makes the system select only stock item records that have *Active* status in the system.

  To define this filter, you will set the properties of the `Filter` object as follows:

  - `Field`: The `IN202500StockItemSummary.ItemStatus` field
  - `Value`: `"Active"`
  - `Condition`: `FilterCondition.Equals`
  - `Operator`: `FilterOperator.And`, which designates that the system should select records that satisfy both this filter and the following filter.

- The filter that makes the system select only stock item records that have been changed within the past month.

  To define this filter, you will set the properties of the `Filter` object as follows:

  - `Field`: The `LastModifiedDateTime` field of the data access class underlying the Summary area of the Stock Items form
  - `Value`: A day before the current date
  - `Condition`: `FilterCondition.Greater`

You will use the `Export()` method of the `Screen` object to retrieve the values of the specified fields of the stock item records. You will specify the parameters of the method as follows:

- Pass the configured array of commands as the first parameter.
- Pass the configured array of filters to the second parameter.

Copyright 2019 MYOB Technology Pty Ltd

- Pass `0` as the third parameter because you do not need to limit the number of records.
- Pass `true` as the fourth parameter to include column headers in the result of the export. The first row of the exported data will include the names of exported elements in the first row.
- Pass `false` as the fifth parameter because you do not need to stop the export if an error occurs during this process.

Retrieving the List of Stock Items

To export the changed stock items, do the following:

1. In the `Integration` folder of the *MyBIIntegrationSBAPI* project, add a new C# class, `RetrievalOfDelta`.
2. In the `RetrievalOfDelta.cs` file, type the `using` directives as shown in the following code to make the `RetrievalOfDelta` class use the *MyBIIntegration* web reference and the classes of `System.IO` and `System.Threading` namespaces.

```
using MyBIIntegrationSBAPI.MyBIIntegration;
using System.Threading;
using System.IO;
```

3. In the `StockItem` class, define the `ExportStockItems()` method, as shown in the following code.

```
//Exporting the list of the stock items that satisfy two conditions
public static void ExportStockItems(Screen screen)
{
    Console.WriteLine("Retrieving the list of stock items...");

    //Specify a locale to make MYOB Advanced ERP handle dates correctly
    screen.SetLocaleName(Thread.CurrentThread.CurrentCulture.ToString());

    //Get the schema of the Stock Items (IN202500) form and
    //configure the sequence of commands
    IN202500Content stockItemsSchema =
      PX.Soap.Helper.GetSchema<IN202500Content>(screen);
    var commands = new Command[]
    {
        //Specify the elements whose values should be exported
        stockItemsSchema.StockItemSummary.ServiceCommands.EveryInventoryID,
        stockItemsSchema.StockItemSummary.InventoryID,
        stockItemsSchema.StockItemSummary.Description,
        stockItemsSchema.StockItemSummary.ItemStatus,
        stockItemsSchema.GeneralSettingsItemDefaults.ItemClass,
        stockItemsSchema.GeneralSettingsUnitOfMeasureBaseUnit.BaseUnit,
        new Field
        {
            ObjectName =
              stockItemsSchema.StockItemSummary.InventoryID.ObjectName,
            FieldName = "LastModifiedDateTime"
        },
        stockItemsSchema.WarehouseDetails.Warehouse,
        stockItemsSchema.WarehouseDetails.QtyOnHand
    };

    //Filter the records to be exported
    var filters = new Filter[]
    {
        //Export only the records that have the Active status
        new Filter
        {
            Field = stockItemsSchema.StockItemSummary.ItemStatus,
            Condition = FilterCondition.Equals,
            Value = "Active",
            Operator = FilterOperator.And
```

Copyright 2019 MYOB Technology Pty Ltd

```
        },
        //And only the records that were modified within the last month
        new Filter
        {
            Field = new Field
            {
                ObjectName =
                  stockItemsSchema.StockItemSummary.InventoryID.ObjectName,
                FieldName = "LastModifiedDateTime"
            },
            Condition = FilterCondition.Greater,
            Value = DateTime.Now.AddDays(-1).ToLongDateString()
        }
    };

    //Export stock item records
    String[][] items =
      screen.IN202500Export(commands, filters, 0, true, false);

    //Save the data to a CSV file
    StreamWriter file = new StreamWriter("StockItems.csv");
    {
        foreach (string[] rows in items)
        {
            foreach (string row in rows)
            {
                file.Write(row + ";");
            }
            file.WriteLine();
        }
    }
    file.Close();
}
```

4 In the `try` block of the `Main()` method of the `Program` class, call the `ExportStockItems()` method of the `RetrievalOfDelta` class, as the following code shows.

```
RetrievalOfDelta.ExportStockItems(screen);
```

5 Rebuild the project, and run the application. You can find the `StockItems.csv` file with the results of the export in the `\bin\Debug` folder of the *MyBIIntegration* project if you run the project in the Debug mode.

The resulting file is shown in the following screenshot. Check that all active stock item records that were modified within the last day were exported, and the inactive *AACOMPUT01* stock item is not in the list.

	A	B	C	D	E	F	G	H
1	InventoryID	Description	ItemStatus	ItemClass	BaseUnit	LastModifiedDateTime	Warehouse	QtyOnHand
2	AALEGO500	Lego, 500 piece set	Active	STOCKITEM	PIECE	14.11.2018 11:06	MAIN	2000
3								

Figure: `StockItems.csv` file

Related Links

Screen-Based API Wrapper
Export() Method
SetLocaleName() Method
Commands for Retrieving the Values of Elements
Selection of a Group of Records for Export

Copyright 2019 MYOB Technology Pty Ltd

Lesson Summary

In this lesson, you have learned how to export from MYOB Advanced ERP the list of stock item records that were modified within a particular period by using different integration approaches.

The following table summarizes the options for the retrieval of the delta of records for each of the integration interfaces.

Integration Interface	Ability to Filter the Data by Conditions
OData interface	Yes
REST API	Yes
Contract-based SOAP API	Yes
Screen-based SOAP API	Yes

Copyright 2019 MYOB Technology Pty Ltd

Lesson 3.2: Monitoring Item Availability with Push Notifications

The BI application of the MyStore company needs to display up-to-date information about the item availability in warehouses. You will configure MYOB Advanced ERP to track the changes in item availability and to send notifications to an HTTP address when the availability of any item changes. (The processing of these notifications in the BI application is outside of the scope of this course.) With push notifications, the BI application does not need to continually poll for the data to find out whether there are any changes to this data, which helps improve the performance of the application.

Lesson Objective

In this lesson, you will learn how to configure MYOB Advanced ERP to send push notifications to an HTTP address.

Copyright 2019 MYOB Technology Pty Ltd

Example 3.2.1: Configuring Push Notifications

In this example, you will configure MYOB Advanced ERP to monitor changes in the Item Availability Data (INGI0002) custom generic inquiry. MYOB Advanced ERP will send push notifications to an HTTP address in JSON format about the changes.

Prerequisites

To test the push notifications that you will set up MYOB Advanced ERP to send to an HTTP address, you will use the *http://webhookinbox.com/* website, which is an open-source service. To obtain an HTTP address that you will use for testing, do the following:

1. Open *http://webhookinbox.com/*.
2. Click **create an inbox**.
3. Copy the HTTP address that the site has generated. An example of the address is shown in the following screenshot. Do not close the webpage; you will use it for testing.

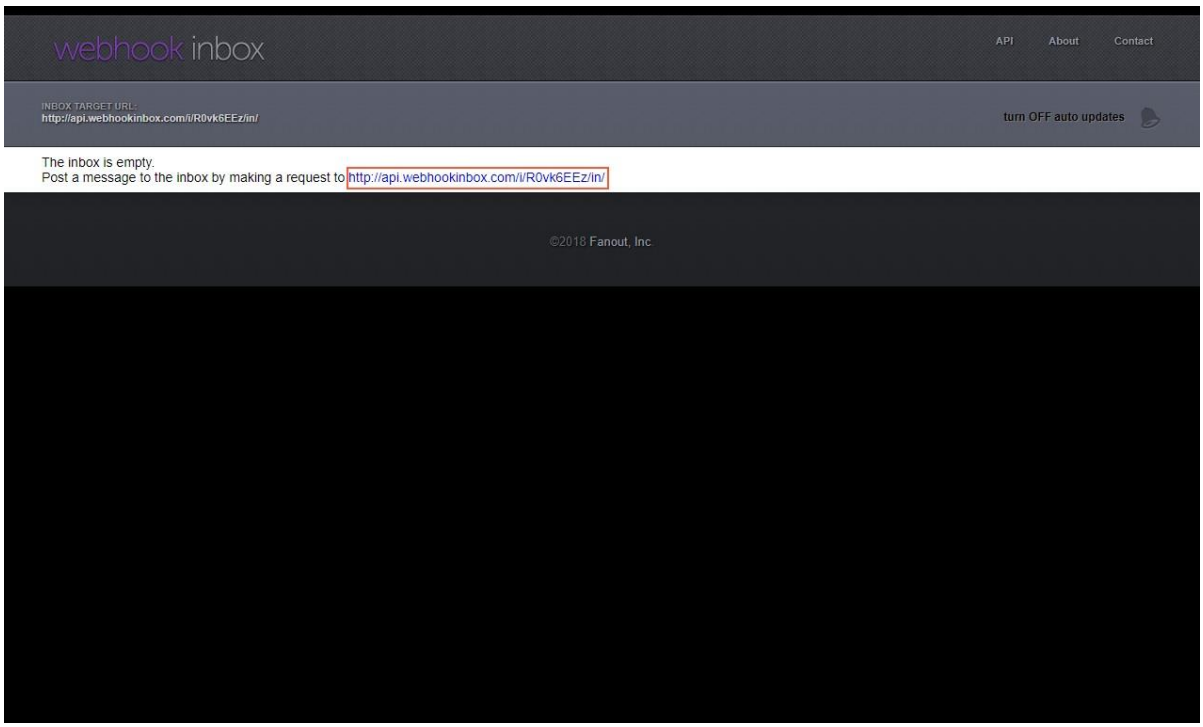


Figure: An HTTP address

Instructions for Configuring Push Notifications

To configure MYOB Advanced ERP to send push notifications to an HTTP address, do the following:

1. In the Summary area of the Push Notifications (SM302000) form, specify the following settings:
   - **Destination Name**: `MyBIIntegration`

     You can specify any name for the target notification destination. In this example, we use the name of the integration application as the destination name.
   - **Destination Type**: *Webhook*

     With *Webhook* selected, MYOB Advanced ERP will send HTTP `POST` requests with notification information to an HTTP address.
   - **Address**: The address that you have created in the prerequisites of this example

Copyright 2019 MYOB Technology Pty Ltd

2. On the **Generic Inquiries** tab, click **Add Row** on the table toolbar, and in the **Inquiry Title** column of the added row, select *Item Availability Data*.
3. On the form toolbar, click **Save**. The following screenshot shows the final configuration.

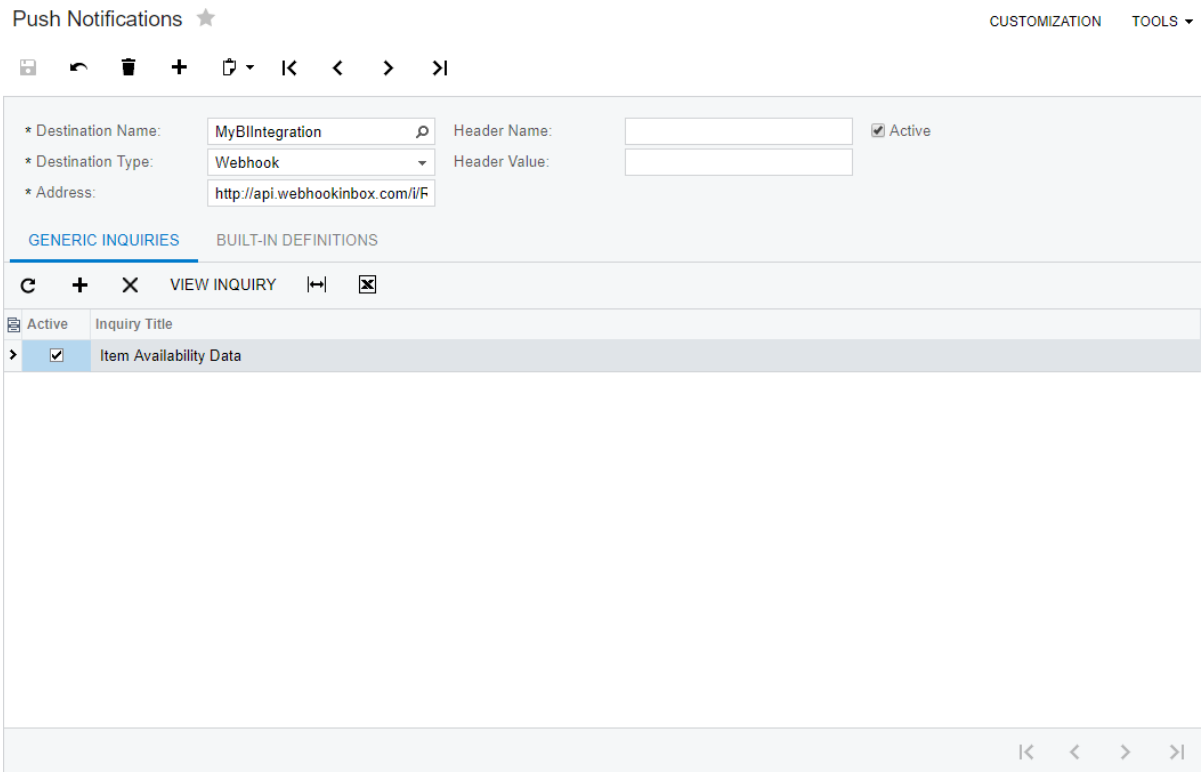


Figure: Push notification configuration

Instructions for Testing Push Notifications

To test the configured notifications, do the following:

1. On the Sales Orders (SO301000), create a sales order as follows:
   a. In the **Customer** box, select *C000000001*.
   b. Clear the **Hold** check box.
   c. On the table toolbar of the **Document Details** tab, click **Add Stock Item**. In the dialog box that opens, select the unlabeled check box for the *AALEGO500* inventory item, and click **Add & Close**.
   d. On the form toolbar, click **Save**.
2. On the webhook inbox webpage, which you have generated in the prerequisites, review the push notification that MYOB Advanced ERP sent for the change in the available quantity for the *AALEGO500* inventory item. When you created a sales order, the available quantity of the *AALEGO500* inventory item was changed. The notification contains the information about the new value of the `QtyAvailable` field of the changed record in the `Inserted` element and the information about the previous value of this field in the `Deleted` element. An example of the notification is shown in the following code.

```
{
  "Inserted":
  [
    {
      "Warehouse":"MAIN ",
```

Copyright 2019 MYOB Technology Pty Ltd

```
      "InventoryID":"AALEGO500",
      "Description":"Lego, 500 piece set",
      "QtyOnHand":2000.000000,
      "QtyAvailable":1974.000000
    }
  ],
  "Deleted":
  [
    {
      "Warehouse":"MAIN",
      "InventoryID":"AALEGO500 ",
      "Description":"Lego, 500 piece set",
      "QtyOnHand":2000.000000,
      "QtyAvailable":1975.000000
    }
  ],
  "Query":"Item Availability Data",
  "CompanyId":"MyStore",
  "Id":"06284bdf-2952-4a9a-bf60-2ad0cd33924e",
  "TimeStamp":636771954942887487,
  "AdditionalInfo":
  {
    "PXPerformanceInfoStartTime":"11/07/2018 13:51:34"
  }
}
```


Related Links

Push Notifications
To Configure Push Notifications
Push Notification Format

Copyright 2019 MYOB Technology Pty Ltd

Additional Information: Push Notifications

The scenarios described in this topic are outside of the scope of this course but may be useful to some readers.

Sending Push Notifications to a Message Queue or the SignalR Hub

MYOB Advanced ERP can send notifications to the following predefined destination types: an HTTP address (webhook), a Microsoft message queue, and the SignalR hub (the destination type implemented in MYOB Advanced ERP by using the ASP.NET SignalR library).

For details about these destination types, see *Push Notification Destinations*. For information about how to configure MYOB Advanced ERP to send notifications to a message queue or the SignalR hub, see *To Configure Push Notifications*.

Processing Failed Notifications

If MYOB Advanced ERP is configured to send push notifications to an HTTP address or to a message queue and cannot send these notifications for some reason, MYOB Advanced ERP saves the information about these notifications and displays these notifications on the Process Push Notifications (SM502000) form. You can resend the failed notifications for two days, after which the notifications are removed from the MYOB Advanced ERP database. For details about how to resend the failed notifications, see *To Process Failed Notifications*.

If MYOB Advanced ERP is configured to send push notifications to the SignalR hub, the failed notifications cannot be resent. For details, see *Push Notification Destinations*.

Defining the Data Query for Push Notifications in Code

You can define the data query that specifies the data changes for which MYOB Advanced ERP should send notifications in one of the following ways:

- By using a generic inquiry, as has been described in this lesson
- By using a built-in definition, which is a data query defined in code

For the information about how to create a built-in definitions, see *To Create a Built-In Definition*.

Adding Custom Information to Push Notifications

Push notifications in JSON format that an external application receives include the `AdditionalInfo` element. In this element, you can include custom information, such as the name of the user that performed the data change. For details about how to include custom information in push notifications, see *To Add Additional Information to Push Notifications*.

Creating a Custom Destination for Push Notifications

MYOB Advanced ERP can send notifications to the following predefined destination types: an HTTP address, a Microsoft message queue, and the SignalR hub. If these destination types do not satisfy the requirements of your application, you can create a custom destination type in code. For details about how to develop a custom destination type, see *To Create a Custom Destination Type*.

Copyright 2019 MYOB Technology Pty Ltd

Lesson Summary

In this lesson, you have configured MYOB Advanced ERP to send push notifications to an HTTP address. You have also reviewed the possible options for the configuration of push notifications.

Copyright 2019 MYOB Technology Pty Ltd

Part 4: Creation of a Customization Package

After the MyBIIntegration application is ready, to distribute the application among the MYOB Advanced ERP instances of the company, you need to include all the custom items that you have created in MYOB Advanced ERP in a customization package.

By completing the examples of this part, you will configure customization packages that include all the items that have been created in MYOB Advanced ERP for each of the integration interfaces.

Copyright 2019 MYOB Technology Pty Ltd

Lesson 4.1: Configuring a Customization Project and Exporting It

In this lesson, for each of the integration interfaces described in this course, you will create a customization project, include all necessary data in this project, and export the project as a ZIP file. You will need to include the following items in the customization project:

- For the integration through OData:
  - All generic inquiries that are used in the integration application
  - The push notification destination
- For the integration through the contract-based REST API or SOAP API:
  - The generic inquiry that is used in the integration application
  - The custom endpoint
  - The push notification destination
- For the integration through the screen-based SOAP API:
  - The generic inquiry that is used in the integration application
  - The push notification destination

Lesson Objectives

In this lesson, you will learn how to do the following:

- Include in a customization project all necessary items that were created in MYOB Advanced ERP for an integration application
- Export the customization project as a ZIP file

Copyright 2019 MYOB Technology Pty Ltd

Example 4.1.1: Creating a Customization Package for the OData Integration

In this example, you will create a customization package for the OData integration that you have implemented during this course.

You will include in the customization package the following items:

- Generic inquiries:
  - Customer Contacts (ARGI0015), which was used in *Example 2.1.1: Using a Custom Generic Inquiry (OData)*
  - Item Availability Data (INGI0002), which was used in *Example 2.2.1: Using a Generic Inquiry (OData)* and *Example 3.2.1: Configuring Push Notifications*
  - Modified Stock Items (INGI0016), which was used in *Example 3.1.1: Filtering the Result of a Generic Inquiry (OData)*
- Push notification destination:
  - MyBIIntegration, which was created in *Example 3.2.1: Configuring Push Notifications*

You can include any other items in the customization project if they are necessary for the integration application.

Creating a Customization Package

To create a customization project and export it to a ZIP file, do the following:

1. On the Customization Projects (SM204505) form, do the following:
   a. On the form toolbar, click **Add Row**.
   b. In the **Project Name** column of the added row, type the name of the project: `MyBIIntegrationOData`.
   c. On the form toolbar, click **Save**
2. In the **Project Name** column, click the *MyBIIntegrationOData* link, which opens the Customization Project Editor for the *MyBIIntegrationOData* customization project.
3. Include the generic inquiries in the customization project as follows:
   a. In the navigation pane of the Customization Project Editor, click **Generic Inquiries** to open the Generic Inquiries page.
   b. On the toolbar of the Generic Inquiries page, click **Add New Record**.
   c. In the **Add Generic Inquiries** dialog box, which opens, select the check boxes in the rows with the following inquiry titles (as shown in the screenshot below):
      - *Customer Contacts*
      - *Item Availability Data*
      - *Modified Stock Items*

Copyright 2019 MYOB Technology Pty Ltd

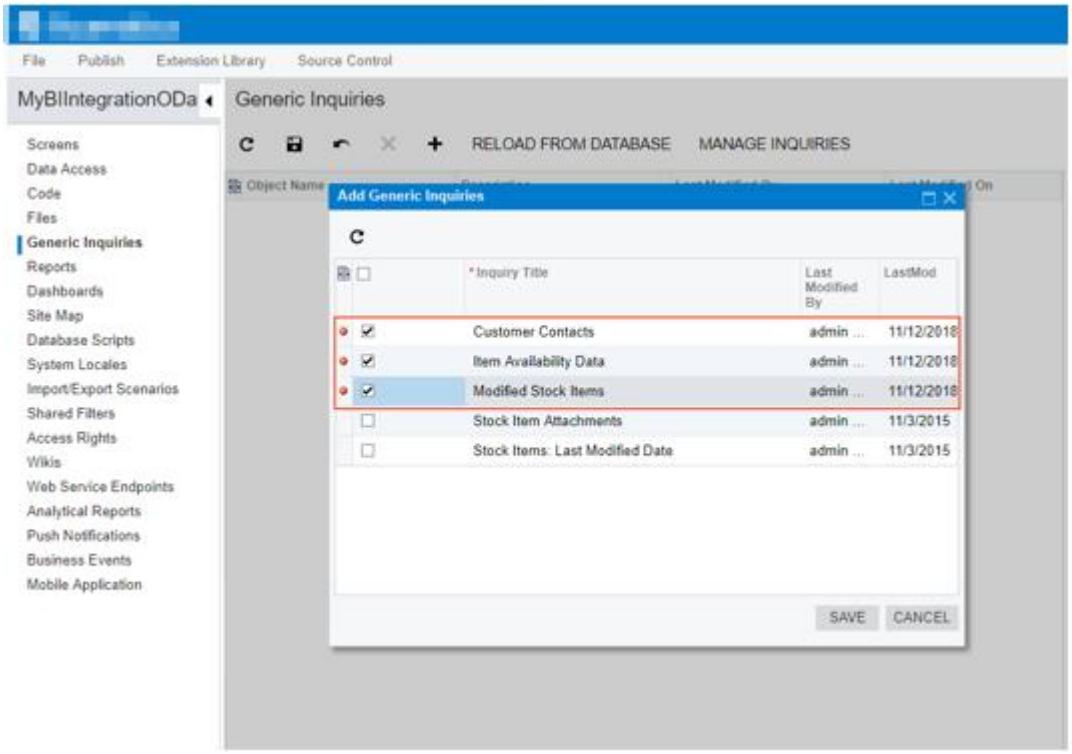


Figure: Selecting the generic inquiries

d. Click **Save**.

   Three inquiries have been added to the Generic Inquiries page.

4. Include the destination of push notifications in the customization project as follows:

   a. In the navigation pane of the Customization Project Editor, click **Push Notifications**.

   b. On the toolbar of the Push Notifications page, click **Add New Record**.

   c. In the **Add Push Notifications** dialog box, select the check box in the row for the *MyBIIntegration* destination name.

   d. Click **Save**.

      The *MyBIIntegration* destination has been added to the Push Notifications page.

5. Close the Customization Project Editor.

6. On the Customization Projects (SM204505) form, click the row for the *MyBIIntegrationOData* customization project, and on the form toolbar, click **Export**; save the ZIP file with the customization project to your computer.

   You can import this ZIP file to another MYOB Advanced ERP instance and publish the customization project to make the items from the customization project available in this instance.

Related Links

Managing Items in a Project

Copyright 2019 MYOB Technology Pty Ltd

Example 4.1.2: Creating a Customization Package for the Contract-Based REST and SOAP API Integration

In this example, you will create a customization package for the contract-based REST and contract-based SOAP API integrations that you have implemented during this course.

You will include in the customization package the following items:

- The *ItemAvailabilityData/0001* custom endpoint, which was used in *Lesson 2.2: Retrieving the Quantities of Stock Items* to retrieve the data from the Item Availability Data (INGI0002) generic inquiry
- The Item Availability Data generic inquiry, which was used in *Lesson 2.2: Retrieving the Quantities of Stock Items* and *Example 3.2.1: Configuring Push Notifications*
- The MyBIIntegration push notification destination, which was created in *Example 3.2.1: Configuring Push Notifications*

You can include any other items in the customization project if they are necessary for the integration application.

Creating a Customization Package

To create a customization project and export it to a ZIP file, do the following:

1. On the Customization Projects (SM204505) form, do the following:
   a. On the form toolbar, click **Add Row**.
   b. In the **Project Name** column of the added row, type the name of the project: `MyBIIntegrationCBAPI`.
   c. On the form toolbar, click **Save**
2. In the **Project Name** column, click the *MyBIIntegrationCBAPI* link, which opens the Customization Project Editor for the *MyBIIntegrationCBAPI* customization project.
3. Include the custom endpoint in the customization project as follows:
   a. In the navigation pane of the Customization Project Editor, click **Web Service Endpoints** to open the Web Service Endpoints page.
   b. On the toolbar of the Web Service Endpoints page, click **Add New Record**.
   c. In the **Add Entity Endpoint** dialog box, which opens, select the check box in the row for the *ItemAvailabilityData* endpoint with Version *0001*, as shown in the following screenshot.

Copyright 2019 MYOB Technology Pty Ltd

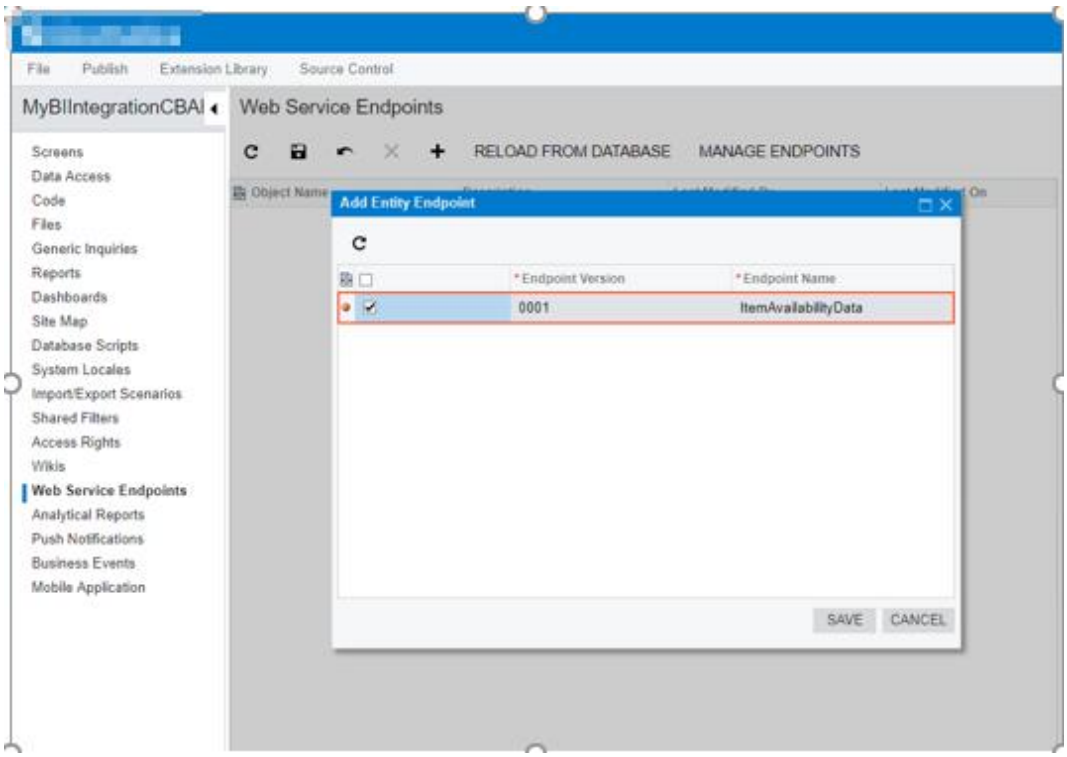


Figure: Selecting the endpoint

d. Click **Save**.

   The endpoint has been added to the Web Service Endpoints page.

4. Include the generic inquiry in the customization project as follows:

   a. In the navigation pane of the Customization Project Editor, click **Generic Inquiries**.

   b. On the toolbar of the Generic Inquiries page, click **Add New Record**.

   c. In the **Add Generic Inquiries** dialog box, select the check box in the row for the *Item Availability Data* generic inquiry.

   d. Click **Save**.

      The inquiry has been added to the Generic Inquiries page.

5. Include the destination of push notifications in the customization project as follows:

   a. In the navigation pane of the Customization Project Editor, click **Push Notifications**.

   b. On the toolbar of the Push Notifications page, click **Add New Record**.

   c. In the **Add Push Notifications** dialog box, select the check box in the row for the *MyBIIntegration* destination name.

   d. Click **Save**.

      The *MyBIIntegration* destination has been added to the Push Notifications page.

6. Close the Customization Project Editor.

7. On the Customization Projects (SM204505) form, click the row for the *MyBIIntegrationCBAPI* customization project, and on the form toolbar, click **Export**; save the ZIP file with the customization project to your computer.

Copyright 2019 MYOB Technology Pty Ltd

You can import this ZIP file to another MYOB Advanced ERP instance and publish the customization project to make the items from the customization project available in this instance.

Related Links

Managing Items in a Project

Copyright 2019 MYOB Technology Pty Ltd

Example 4.1.3: Creating a Customization Package for the Screen-Based SOAP API Integration

In this example, you will create a customization package for the screen-based SOAP API integration that you have implemented during this course.

You will include in the customization package the following items:

- The Item Availability Data (INGI0002) generic inquiry, which was used in *Example 2.2.4: Using the Export() Method to Retrieve Data from an Inquiry (Screen-Based SOAP)* and *Example 3.2.1: Configuring Push Notifications*
- The MyBIIntegration push notification destination, which was created in *Example 3.2.1: Configuring Push Notifications*

You can include any other items in the customization project if they are necessary for the integration application.

Creating a Customization Package

To create a customization project and export it to a ZIP file, do the following:

1. On the Customization Projects (SM204505) form, do the following:
    a. On the form toolbar, click **Add Row**.
    b. In the **Project Name** column of the added row, type the name of the project: `MyBIIntegrationSBAPI`.
    c. On the form toolbar, click **Save**
2. In the **Project Name** column, click the *MyBIIntegrationSBAPI* link, which opens the Customization Project Editor for the *MyBIIntegrationSBAPI* customization project.
3. Include the generic inquiry in the customization project as follows:
    a. In the navigation pane of the Customization Project Editor, click **Generic Inquiries** to open the Generic Inquiries page.
    b. On the toolbar of the Generic Inquiries page, click **Add New Record**.
    c. In the **Add Generic Inquiries** dialog box, which opens, select the check box in the row for the *Item Availability Data* generic inquiry, as shown in the following screenshot.

Copyright 2019 MYOB Technology Pty Ltd

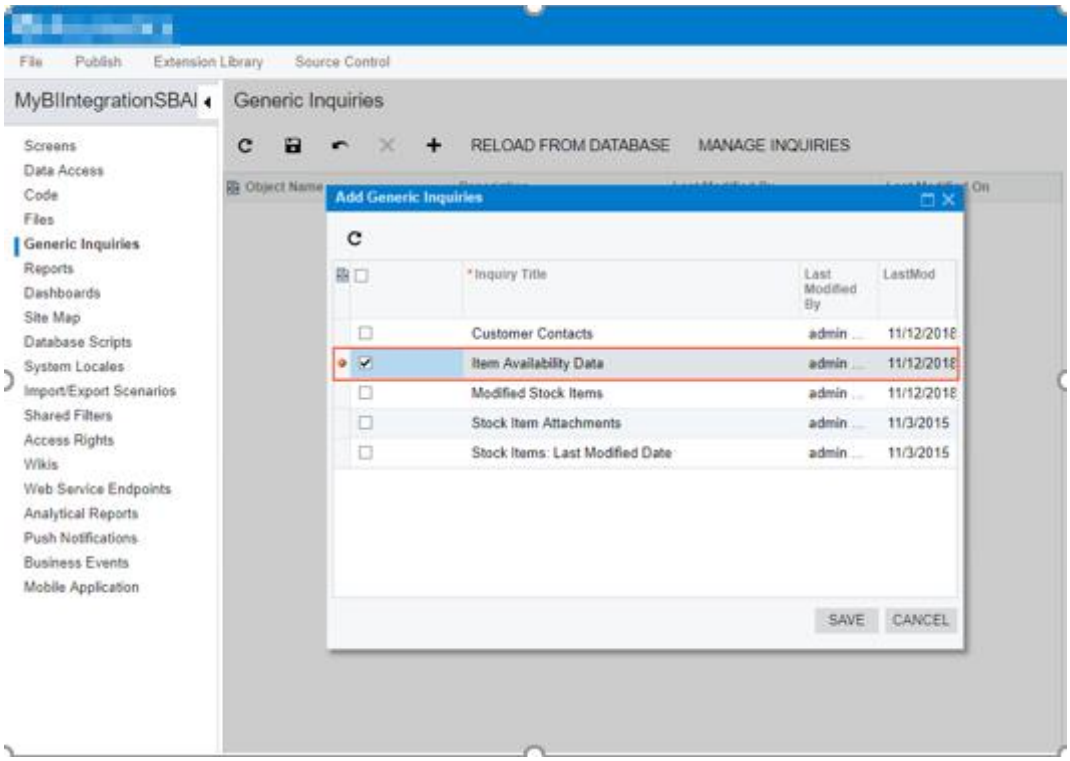


Figure: Selecting the generic inquiry

d. Click **Save**.

   The inquiry has been added to the Generic Inquiries page.

4. Include the destination of push notifications in the customization project as follows:

   a. In the navigation pane of the Customization Project Editor, click **Push Notifications**.

   b. On the toolbar of the Push Notifications page, click **Add New Record**.

   c. In the **Add Push Notifications** dialog box, select the check box in the row for the *MyBIIntegration* destination name.

   d. Click **Save**.

      The *MyBIIntegration* destination has been added to the Push Notifications page.

5. Close the Customization Project Editor.

6. On the Customization Projects (SM204505) form, click the row for the *MyBIIntegrationSBAPI* customization project, and on the form toolbar, click **Export**; save the ZIP file with the customization project to your computer.

   You can import this ZIP file to another MYOB Advanced ERP instance and publish the customization project to make the items from the customization project available in this instance.

Related Links

Managing Items in a Project

Copyright 2019 MYOB Technology Pty Ltd

Lesson Summary

In this lesson, you have configured a customization project to include all necessary items that have been created in MYOB Advanced ERP for the integration application. You have exported this customization project as a ZIP file.

The following table summarizes which items should be included in the customization projects for each of the integration interfaces.

Integration Interface	Generic Inquiries	Web Service Endpoints	Push Notifications	Other Customization Project Items
OData interface	Yes	No	Yes, if push notifications have been used for the integration	Yes, if these items have been used for the integration
REST API	Yes, if generic inquiries have been used in a custom endpoint or an endpoint extension	Yes, if a custom endpoint or an endpoint extension is used	Yes, if push notifications have been used for the integration	Yes, if these items have been used for the integration
Contract-based SOAP API	Yes, if generic inquiries have been used in a custom endpoint or an endpoint extension	Yes, if a custom endpoint or an endpoint extension is used	Yes, if push notifications have been used for the integration	Yes, if these items have been used for the integration
Screen-based SOAP API	Yes, if generic inquiries have been used for the integration	No	Yes, if push notifications have been used for the integration	Yes, if these items have been used for the integration

Copyright 2019 MYOB Technology Pty Ltd

Appendix: Comparison of the Integration Interfaces

This topic summarizes the differences between the integration approaches described in this training guide.

Initial Configuration

The following table summarizes the configuration of MYOB Advanced ERP and the integration application that should be performed for each of the integration interfaces.

Integration Interface	Configuration of MYOB Advanced ERP	Configuration of the Integration Application
OData interface	Optional: Configuration of CORS	No
REST API	Optional: Configuration of a custom endpoint or an endpoint extension	No
Contract-based SOAP API	Optional: Configuration of a custom endpoint or endpoint extension	• Import of the WSDL description of the web service • Configuration of the integration application to use cookies
Screen-based SOAP API	Optional: Configuration of the WSDL description of the web service	Import of the WSDL description of the web service

Signing In to and Signing Out from MYOB Advanced ERP

The following table summarizes the sign-in and sign-out options and limitations for each of the integration interfaces.

Integration Interface	Authentication and Authorization	Limit for the Number of API Users
OData interface	Basic authentication	No
REST API	• `POST` requests to the sign-in and sign-out endpoints • OAuth 2.0 authorization	Yes
Contract-based SOAP API	• The `Login()` and `Logout()` of a `DefaultSoapClient` object • OAuth 2.0 authorization	Yes
Screen-based SOAP API	The `Login()` and `Logout()` of a `Screen` object	Yes

Data Retrieval from a Data Entry Form

The following table summarizes the options of data retrieval from a data entry form for each of the integration interfaces.

Copyright 2019 MYOB Technology Pty Ltd

Integration Interface	Ability to Retrieve Data from a Data Entry Form	Ability to Retrieve Custom Fields
OData interface	No. You need to create a custom generic inquiry that retrieves the needed data.	Yes
REST API	Yes, if the form is mapped in an endpoint.	Yes
Contract-based SOAP API	Yes, if the form is mapped in an endpoint.	Yes
Screen-based SOAP API	Yes.	Yes

Data Retrieval from a Generic Inquiry

The following table summarizes the options of data retrieval from a generic inquiry form for each of the integration interfaces.

Integration Interface	Ability to Retrieve Data from a Generic Inquiry Without Parameters	Ability to Specify the Values of Parameters of Generic Inquiries
OData interface	Yes	No
REST API	Yes, if the generic inquiry is mapped in an endpoint	Yes, if the generic inquiry and its parameters are mapped in an endpoint
Contract-based SOAP API	Yes, if the generic inquiry is mapped in an endpoint	Yes, if the generic inquiry and its parameters are mapped in an endpoint
Screen-based SOAP API	Yes	Yes

Retrieval of the Delta of Records

The following table summarizes the options for the retrieval of the delta of records for each of the integration interfaces.

Integration Interface	Ability to Filter the Data by Conditions
OData interface	Yes
REST API	Yes
Contract-based SOAP API	Yes
Screen-based SOAP API	Yes

Creation of the Customization Package

The following table summarizes which items should be included in the customization projects for each of the integration interfaces.

Copyright 2019 MYOB Technology Pty Ltd

Integration Interface	Generic Inquiries	Web Service Endpoints	Push Notifications	Other Customization Project Items
OData interface	Yes	No	Yes, if push notifications have been used for the integration	Yes, if these items have been used for the integration
REST API	Yes, if generic inquiries have been used in a custom endpoint or an endpoint extension	Yes, if a custom endpoint or an endpoint extension is used	Yes, if push notifications have been used for the integration	Yes, if these items have been used for the integration
Contract-based SOAP API	Yes, if generic inquiries have been used in a custom endpoint or an endpoint extension	Yes, if a custom endpoint or an endpoint extension is used	Yes, if push notifications have been used for the integration	Yes, if these items have been used for the integration
Screen-based SOAP API	Yes, if generic inquiries have been used for the integration	No	Yes, if push notifications have been used for the integration	Yes, if these items have been used for the integration

Copyright 2019 MYOB Technology Pty Ltd

Appendix: Web Integration Scenario Reference

In this topic, you can find reference links to the topics that describe how to implement the following web integration scenarios:

- **Monitoring item availability in warehouses**: *Lesson 3.2: Monitoring Item Availability with Push Notifications*.
- **Retrieving the list of customers**: *Lesson 2.1: Retrieving the List of Customers with Contacts*.
- **Retrieving the list of modified stock items**: *Lesson 3.1: Retrieving the List of Modified Stock Items*.
- **Retrieving the quantities of stock items**: *Lesson 2.2: Retrieving the Quantities of Stock Items*.
- **Signing in**: *Lesson 1.2: Signing In to and Signing Out from MYOB Advanced ERP*.
- **Signing out**: *Lesson 1.2: Signing In to and Signing Out from MYOB Advanced ERP*.

Copyright 2019 MYOB Technology Pty Ltd

Appendix: Troubleshooting

I get the error *API Login Limit* when I try to sign in to MYOB Advanced ERP through the REST API, contract-based SOAP API, or screen-based SOAP API. What should I do?

For an application that uses the API methods for the sign-in to MYOB Advanced ERP (that is, uses cookies), this error appears if all of the following is true:

- The API login limit is specified in the MYOB Advanced ERP license. (The license restriction for the API users is shown in the **Maximum Number of Web Services API Users** box on the **License** tab of the License Maintenance (SM201510) form.)
- The number of unclosed sessions (that is, the sessions in which you have signed in to MYOB Advanced ERP through one of the web services API or obtained access to MYOB Advanced ERP web services APIs through OAuth 2.0 and have not signed out from MYOB Advanced ERP) equals the API login limit in the license.
- You try to sign in to MYOB Advanced ERP once more in another

session. You can deal with this error as follows:

1. Modify the code of your application so that it closes the session (signs out from MYOB Advanced ERP) each time the work with MYOB Advanced ERP is finished. For details, see *Lesson 1.2: Signing In to and Signing Out from MYOB Advanced ERP*.
2. If, for some reason, the integration application has not closed the session, you can do one of the following:
    - Sign out from MYOB Advanced ERP by using the cookies that the application used during the signing in.
    - Wait for ten minutes until the session that is managed by cookies expires.
    - Restart the site in the Internet Information Services (IIS) Manager or by clicking the **Restart Application** button on the toolbar of the Apply Updates (SM203510) form.

I get the error *HTTP 404 Not Found* when I click View Endpoint Service on the Web Service Endpoints (SM207060) form. What should I do?

This error can appear if the MYOB Advanced ERP website is running on a 64-bit operating system, uses the *Classic* managed pipeline mode for the application pool, and uses settings for this application pool that are not recommended. Do the following to make sure you have the recommended IIS settings:

1. In the Control Panel, open **Administrative Tools** > **Internet Information Services (IIS) Manager**.
2. In **Application Pools**, select the application pool that your MYOB Advanced ERP application uses, and click **Advanced Settings** in the **Actions** pane.
3. In the **Advanced Settings** dialog box, which opens, make sure the **Enable 32-Bit Applications** setting is *False*.

Copyright 2019 MYOB Technology Pty Ltd