Developer Course

Web Services

I320 Advanced Data Retrieval with REST API 2022 R1

Revision: 3/23/2022

Acumatica
The Cloud ERP

Contents

Copyright....4
Introduction....5
How to Use This Course....6
Course Prerequisites....7
Deploying an Acumatica ERP Instance for the Training Course....7
Configuring a Website for HTTPS....8
Company Story and MyStoreIntegration Application....11
Part 1: Authorization of the Application to Work with the Web Service....12
Lesson 1.1: Registering the Application in Acumatica ERP....12
Example: Registering the Application in Acumatica ERP....12
Additional Information: OAuth 2.0 Authorization....13
Lesson Summary....13
Lesson 1.2: Configuring the Application to Use OAuth 2.0....14
Example: Configuring the REST Application to Use OAuth 2.0....14
Lesson Summary....15
Lesson 1.3: Signing Out from Acumatica ERP....16
Example: Using the REST API....16
Additional Information: Session Management in the OAuth 2.0 Applications....17
Lesson Summary....17
Part 2: Performance Optimization....18
Lesson 2.1: Retrieving a List of Sales Orders with Details and Related Shipments....18
Example: Using One GET Request....20
Additional Information: Retrieval of the List of Sales Orders Through OData....22
Lesson Summary....22
Lesson 2.2: Retrieving a List of Sales Orders in Batches....23
Example: Using $top and $skip....23
Additional Information: Retrieval of the List of Sales Orders in Batches Through OData....26
Lesson Summary....26
Lesson 2.3: Retrieving the List of Payments One by One....26
Example: Using GET with Key Field Values....27
Lesson Summary....31
Part 3: Retrieval of Attachments....32
Lesson 3.1: Retrieving the Attachments of a Stock Item....32
Prerequisites....32

Example: Using the GET Method..... 33

Lesson Summary..... 34

Appendix: Comparison of the Integration Interfaces.....35

Appendix: Web Integration Scenario Reference.....36

Appendix: Troubleshooting.....37

Copyright

3933 Lake Washington Blvd NE, # 350, Kirkland, WA 98033

Restricted Rights

The product is provided with restricted rights. Use, duplication, or disclosure by the United States Government is subject to restrictions as set forth in the applicable License and Services Agreement and in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (c)(1) and (c)(2) of the Commercial Computer Software-Restricted Rights at 48 CFR 52.227-19, as applicable.

Disclaimer

Acumatica, Inc. makes no representations or warranties with respect to the contents or use of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Acumatica, Inc. reserves the right to revise this document and make changes in its content at any time, without obligation to notify any person or entity of such revisions or changes.

Trademarks

Acumatica is a registered trademark of Acumatica, Inc. HubSpot is a registered trademark of HubSpot, Inc. Microsoft Exchange and Microsoft Exchange Server are registered trademarks of Microsoft Corporation. All other product names and services herein are trademarks or service marks of their respective companies.

Software Version: 2022 R1

Last Updated: 03/23/2022

Introduction

External systems can use Acumatica ERP integration interfaces to access the business functionality and data of Acumatica ERP. Acumatica ERP provides 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 shows advanced techniques of the data retrieval with the contract-based REST API. This course does not cover the implementation of the integration scenarios with the other interfaces. The basic information about the data retrieval through the contract-based REST API is available in the *I310 Data Retrieval with REST API Basics* training course, which is a prerequisite for this course. The information about the contract-based and screen-based SOAP API is available in the *Integration Development Guide*. The basic information about the data retrieval through the OData interface is available in the *I300 Data Retrieval with OData* training course.

The course is intended for developers who need to create applications that interact with Acumatica ERP.

The course is based on a set of examples of web integration scenarios that demonstrate the processes involved in developing a client application that uses Acumatica ERP integration interfaces. The course gives you ideas about how to develop your own applications by using the contract-based REST API.

After you complete all the lessons of the course, you will be familiar with the advanced techniques of data retrieval through the Acumatica ERP contract-based REST API.

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 *Company Story and MyStoreIntegration Application*.
2. Complete the lessons in all parts of the training guide. In each lesson, you should review the description of the lesson, the examples for the integration interfaces, and the lesson summary.
3. In Partner University, take *I320 Certification Test: Data Retrieval Advanced*.

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

What Is in a Part?

The first part of the course explains how to authorize a third-party application to work with the Acumatica ERP integration services.

The other 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 Acumatica ERP.

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 contract-based REST API. 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.

What Are the Documentation Resources?

The complete Acumatica ERP and Acumatica Framework documentation is available on *https://help.acumatica.com/* and is included in the Acumatica ERP instance. While viewing any form used in the course, you can click the **Open Help** button in the top pane of the Acumatica ERP screen to bring up a form-specific Help menu; you can use the links on this menu to quickly access form-related information and activities and to open a reference topic with detailed descriptions of the form elements.

Licensing Information

For the educational purposes of this course, you use Acumatica ERP under the trial license, which does not require activation and provides all available features. For the production use of the Acumatica ERP functionality, an administrator has to activate the license the organization has purchased. Each particular feature may be subject to additional licensing; please consult the Acumatica ERP sales policy for details.

Course Prerequisites

To complete this course, you should be familiar with the basic principles of data retrieval with the Acumatica ERP integration services. We recommend that you complete the *I310 Data Retrieval with REST API Basics* training course before you begin this course.

You need to perform the prerequisite actions described in this part before you start to complete the course.

1. Make sure the environment that you are going to use for the training course conforms to the *System Requirements for Acumatica ERP 2022 R1*.
2. Make sure that the Web Server (IIS) features that are listed in *Configuring Web Server (IIS) Features* are turned on.
3. Deploy an instance of Acumatica ERP 2022 R1 with the name *MyStoreInstance* and a tenant that contains the *I100* data. If you have completed the *I310 Data Retrieval with REST API Basics* training course, you can use the instance that you have deployed for this course. For information on how to deploy the instance for the training course, see *Deploying an Acumatica ERP Instance for the Training Course*.
4. Make sure the Postman application is installed on your computer. To download and install Postman, follow the instructions on *https://www.postman.com/downloads/*.
5. Make sure you have HTTP access from the computer where you work with the examples to the Acumatica ERP instance.
6. Because in the examples of this course you will use the OAuth 2.0 authorization, configure HTTPS on the Acumatica ERP website, as described in *Configuring a Website for HTTPS*. If you do not want to use OAuth 2.0 authorization, your application can use the API methods for the sign-in and sign-out (which were described in the *I310 Data Retrieval with REST API Basics* training course) and interact with Acumatica ERP via HTTP.

Deploying an Acumatica ERP Instance for the Training Course

Instead of deploying a new instance, you can use the Acumatica ERP instance that you have deployed for the *I310 Data Retrieval with REST API Basics* training course.

You deploy an Acumatica ERP instance and configure it as follows:

1. Open the Acumatica ERP Configuration Wizard, and deploy a new application instance as follows:
   a. On the **Database Configuration** page of the Acumatica 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
      - **Insert Data**: *I100*
      - **Parent Tenant ID**: 1
      - **Visible**: Selected

   The system creates a new Acumatica 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 Acumatica 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.

Configuring a Website for HTTPS

In the examples of this guide, you will use the secure connection between the API client application and Acumatica ERP.

A secure connection between the client application and the Acumatica ERP website with a Secure Socket Layer (SSL) certificate is required for the authorization of the client application through OAuth 2.0. Therefore, you have to set up the Acumatica ERP website for HTTPS, as described in this topic.

As the Microsoft IIS documentation states, the steps for configuring SSL for a site include the following:

1. You obtain an appropriate certificate. (For the purposes of completing the course, you can create a self-signed server certificate.)
2. You create an SSL binding on a site.
3. You test the website by making a request to the site.
4. Optional: You configure the SSL options.

To complete the examples of this guide, you should create a self-signed certificate and configure SSL binding as follows:

1. Create a self-signed certificate by doing the following:
   a. In the Control Panel, open **Administrative Tools > Internet Information Services (IIS) Manager**.
   b. In the **Features View**, double-click **Server Certificates**.

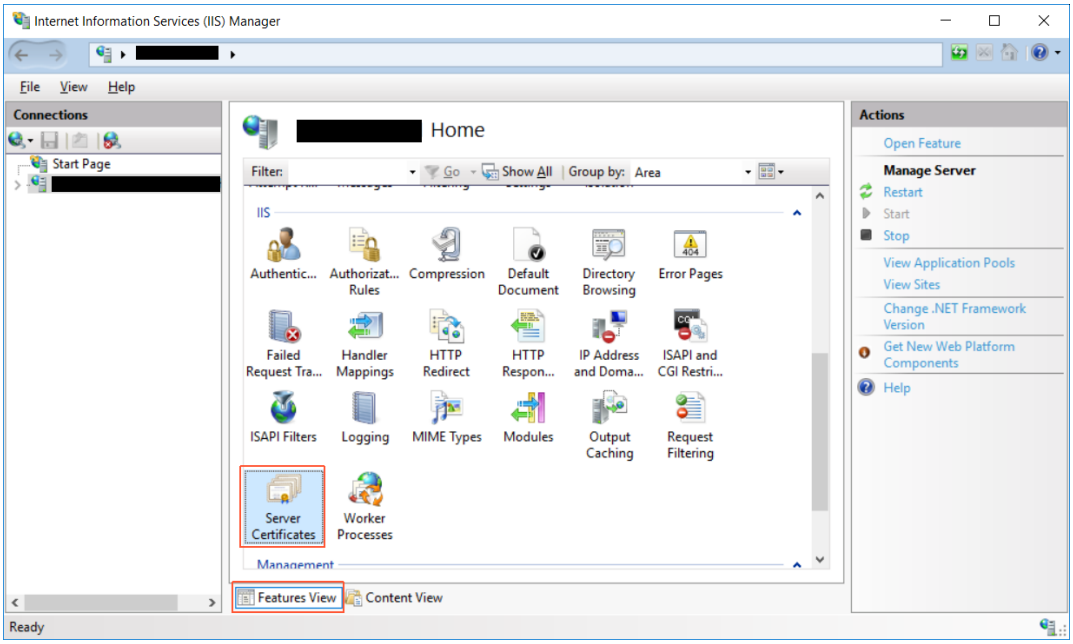

Figure: IIS Manager. Server Certificates icon

   c. Click **Create Self-Signed Certificate** in the **Actions** pane.

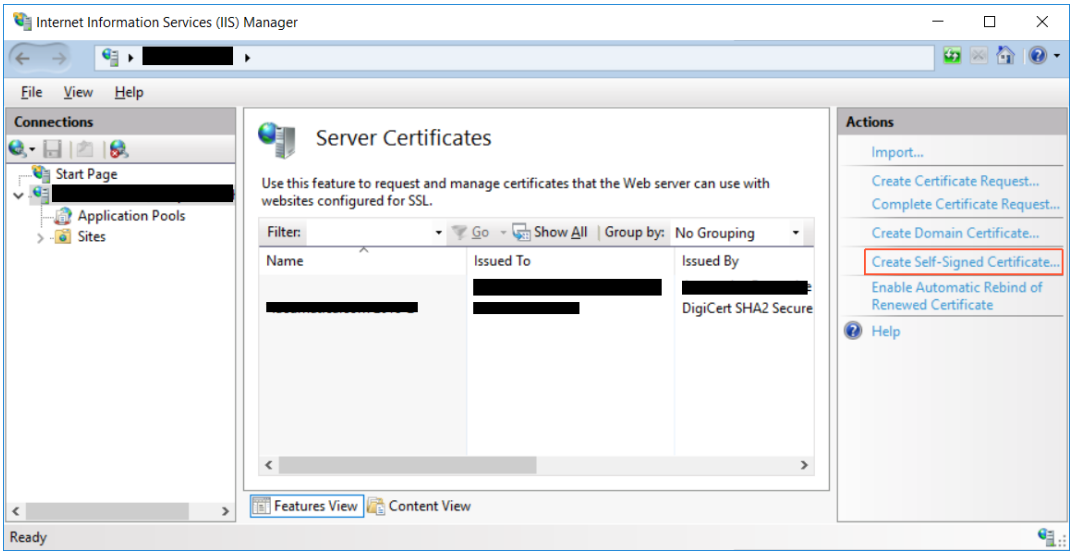

Figure: IIS Manager. Create Self-Signed Certificate link

d. Enter a name for the new certificate, and click **OK**.

2. Do the following to create an SSL binding:

a. Select a site in the tree view, and click **Bindings** in the **Actions** pane.

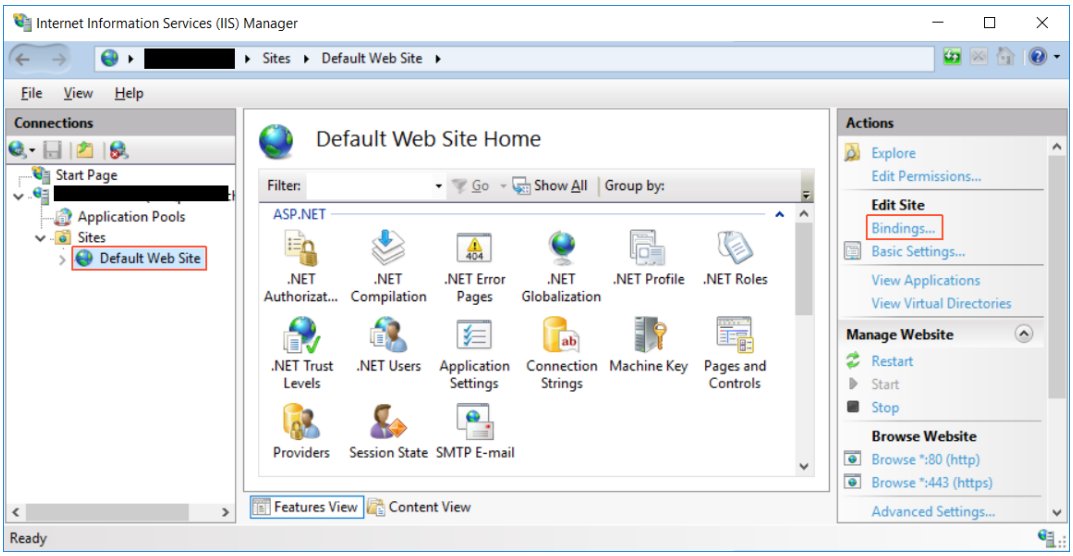

Figure: IIS Manager. Bindings link

b. In the **Site Bindings** dialog box, click **Add** to add your new SSL binding to the site.

c. In the **Type** drop-down list, select *https*.

d. Select the self-signed certificate you created, and click **OK** to close the dialog box.

3. In the **Actions** pane, under **Browse Web Site**, click the link associated with the binding you just created (*Browse *:443 (https)*).

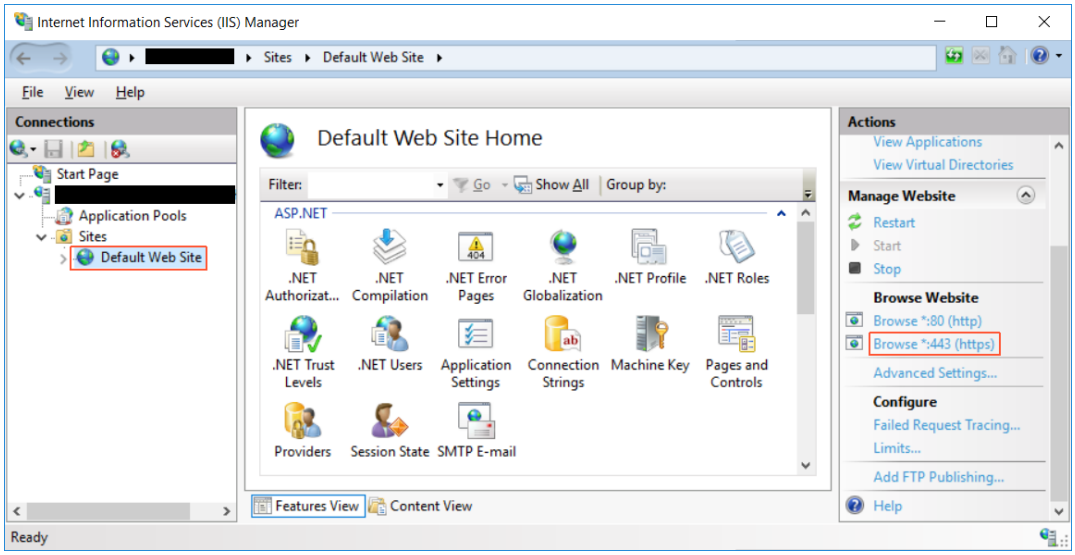

Figure: Opening the HTTPS website

4. Click the link to proceed with this website and disregard the error. The HTTPS website opens.

Company Story and MyStoreIntegration Application

In this course, you will simulate the integration of Acumatica ERP with the online store of a small retail company, MyStore. This company is a single business entity that has no branches or subsidiaries. MyStore uses Acumatica ERP for customer management, sales order processing, and payment collection.
MyStore plans to extend its business and start selling goods online. MyStore needs to investigate the options available in Acumatica ERP for integration with eCommerce applications. In the first stage of implementation, the integration application, which MyStore is developing, should retrieve information about stock items, sales orders, and payments from Acumatica ERP. In the second stage of implementation, the integration application should submit information about customers, sales orders, and payments from the online store to Acumatica ERP. This course covers only the first stage of implementation. The second stage is covered in the *I330 Data Manipulation with REST API* training course.

The MyStoreIntegration application, which you will build as you complete the course, will integrate Acumatica ERP with the online store of the MyStore company. For the implementation of the MyStoreIntegration application, the MyStore company can use the contract-based REST API.

The OData interface can be used only for the implementation of the data retrieval part of the MyStoreIntegration application, while the data submission should be performed by other integration interfaces because the data submission is not possible through OData. This implementation is outside of the scope of this course.

The examples of this course show the implementation of the MyStoreIntegration application with the contract-based REST API.

The following diagram shows how the MyStoreIntegration integration application fits in the integration of the MyStore online store with Acumatica ERP.

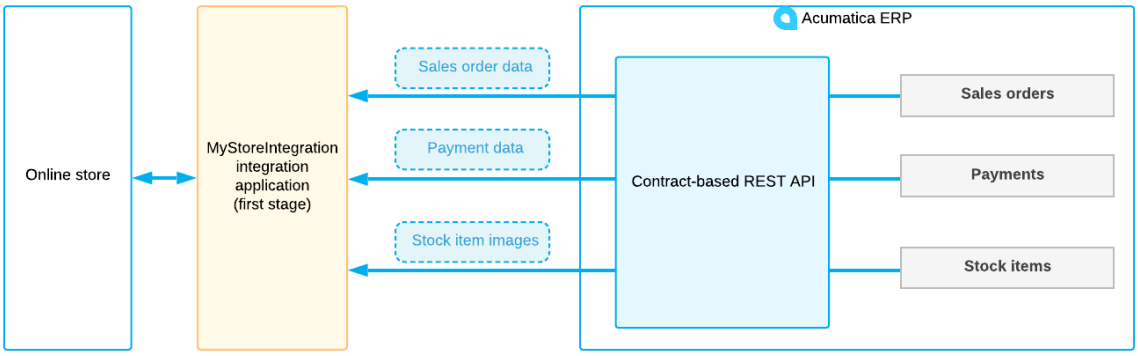

Figure: Integration of the MyStore online store and Acumatica ERP

Integration Requirements

At the first stage of implementation, the MyStoreIntegration application should implement integration with Acumatica ERP to support the following usage scenarios in the online store:

- A registered customer should be able to view all of this customer's purchases.
- A registered customer should be able to view all of this customer's payments.
- A potential customer should be able to view the image of any selected item.

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

Part 1: Authorization of the Application to Work with the Web Service

Before an application can retrieve data from Acumatica ERP by using the integration interfaces provided by Acumatica ERP, the application needs to sign in to Acumatica ERP. You can implement the signing in to and signing out from Acumatica ERP by using the authentication methods of the corresponding integration interfaces. (These methods are described in the *I310 Data Retrieval with REST API Basics* training course and in the documentation.)

Alternatively, you can authorize the application to work with the integration interfaces by using the OAuth 2.0 authorization mechanism. With OAuth 2.0, the client application will not operate with the Acumatica ERP credentials to sign a user in to Acumatica ERP; instead, the application will obtain an access token from Acumatica ERP and will use this token when it requests data from Acumatica ERP.

In this part of the guide, you will register the MyStoreIntegration application in Acumatica ERP and configure the application to use the OAuth 2.0 mechanism of authorization. The MyStoreIntegration application will use the resource owner password credentials flow. With this flow, the credentials (username and password) of an Acumatica ERP user are provided directly to the client application, which uses the credentials to obtain the access token. For details about this flow, see *Resource Owner Password Credentials Flow* in the documentation.

According to the OAuth 2.0 specification, a secure connection between an OAuth 2.0 client application and the Acumatica ERP website with a Secure Socket Layer (SSL) certificate is required. You have set up the Acumatica ERP website for HTTPS before you started this course (as described in *Configuring a Website for HTTPS*).

Lesson 1.1: Registering the Application in Acumatica ERP

In this lesson, you will register the MyStoreIntegration application in Acumatica ERP as a connected application that uses the OAuth 2.0 authorization. To register this application, you will use the *Connected Applications* (SM303010) form. The application will use the resource owner password credentials flow.

When you are registering the client application, you have to be signed in to the tenant whose data the client application needs to access, because the client ID that is generated during the application registration includes the name of the tenant. In the instance that you use for the training course, there is only one tenant configured (with the name *MyStore*).

Lesson Objective

In this lesson, you will learn how to register a client application in Acumatica ERP.

Example: Registering the Application in Acumatica ERP

In this example, you will register the MyStoreIntegration application on the *Connected Applications* (SM303010) form.

Registering a Connected Application

Proceed as follows:

1. In the Summary area of the Connected Applications (SM303010) form, specify the following values:
   - **Client Name**: `MyStoreIntegration`

- **OAuth 2.0 Flow**: *Resource Owner Password Credentials*

2. On the **Secrets** tab, click **Add Shared Secret**.
3. In the **Add Shared Secret** dialog box, which opens, do the following:
   a. In the **Description** box, type `MyStoreIntegration Secret`.
   b. Copy and save the value from the **Value** box.

   > ⚠ For security reasons, the value of the secret is displayed only once: when you create the secret by invoking this dialog box. Therefore, if you do not save the secret, you will not be able to obtain its value in the future.

   c. Click **OK**.
4. On the form toolbar, click **Save**. Notice that the client ID has been generated and inserted in the **Client ID** box. The name of the tenant to which you are signed in is appended to this client ID. The MyStoreIntegration application will use this client ID along with the client secret for authentication in Acumatica ERP.

Related Links

- *To Register a Client Application*

Additional Information: OAuth 2.0 Authorization

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

Use of the Authorization Code Flow and the Implicit Flow

Instead of the resource owner password credential flow, a client application that implements the OAuth 2.0 authorization mechanism can use one of the following OAuth 2.0 authorization flows supported by Acumatica ERP:

- Authorization code: With this authorization flow, the client application never gets the credentials of the applicable Acumatica ERP user. After the user is authenticated in Acumatica ERP, the client application receives an authorization code, exchanges it for an access token, and then uses the access token to work with data in Acumatica ERP. For details on the flow, see *Authorization Code Flow*.
- Implicit: With the implicit flow, the client application never gets the credentials of the applicable Acumatica ERP user. When the user is authenticated in Acumatica ERP, the client application does not receive an authorization code (as with the authorization code flow); instead, the client application directly receives an access token, and then uses the access token to work with data in Acumatica ERP. For more information about the flow, see *Implicit Flow*.

For a comparison of all supported flows, see *Comparison of the Flows*.

Revoking of the Access of a Connected Application

You can revoke the access of a connected application that you have registered on the Connected Applications (SM303010) form. For details about how to do this, see *To Revoke the Access of a Connected Application*.

Lesson Summary

In this lesson, you have learned how to register in Acumatica ERP a client application that uses the OAuth 2.0 authorization. During the registration, you have been signed in to the MyStore tenant, whose data the MyStoreIntegration application needs to access.

You have also reviewed the possible options of the OAuth 2.0 authorization.

The following table summarizes the availability of the OAuth 2.0 authorization method for each of the integration interfaces.

Integration Interface	OAuth 2.0 Authorization
OData Version 3.0 interface	Yes
OData Version 4.0 interface	Yes
REST API	Yes

Lesson 1.2: Configuring the Application to Use OAuth 2.0

In this lesson, you will configure a Postman collection to use OAuth 2.0.

You will connect to the token endpoint, pass the client ID and client secret in the authorization header, and request access to the web service APIs (that is, you will request the `api` scope). You will receive the access token from Acumatica ERP to use it in subsequent requests to Acumatica ERP. You will not request the refresh token, which the client application can use to request a new access token when the access token has expired.

Lesson Objective

In this lesson, you will learn how to configure an integration application to use OAuth 2.0 for authorization in Acumatica ERP.

Example: Configuring the REST Application to Use OAuth 2.0

In this example, you will configure a Postman collection to use the OAuth 2.0 authorization for the requests to Acumatica ERP.

You will connect directly to the token endpoint. In Postman, you cannot use the discovery endpoint, which is *https://<Acumatica ERP instance URL>/identity/*, because Postman does not support OpenID Connect Discovery. (In the discovery endpoint address, *<Acumatica ERP instance URL>* is the URL of the Acumatica ERP instance to which the client application is going to connect.) If the client application supports OpenID Connect Discovery, we recommend that the client application use the discovery endpoint address to obtain the token endpoint address. The use of the discovery endpoint eliminates the need to change the application if the address of the token endpoint changes.

A request to the discovery endpoint does not provide the access token; it provides the address of the token endpoint from which you can receive the access token.

You will configure the Postman collection to request the `api` access scope, which provides access to the web services API. You will not use the `offline_access` scope, which requests that a refresh token be granted.

Configuring a Postman Collection

To configure a Postman collection to use the OAuth 2.0 authorization in Acumatica ERP, do the following:

1. If you use a self-signed certificate for HTTPS, in Postman settings, turn off SSL certificate verification.
2. In Postman, create a collection.

> ⚠ Instead of creating a collection, you can import to Postman the collection provided with this course (`REST.postman_collection.json`). This collection has been configured for this course and already contains all the requests that are used in the course. You can use this collection for testing the requests. (You can find the file in *https://github.com/Acumatica/Help-and-Training-Examples/tree/2022R1/IntegrationDevelopment/I320*.)

3. On the **Authorization** tab of the collection properties window, which opens when the collection has been created, do the following:
   a. Select the following values:
      - **Type**: *OAuth 2.0*
      - **Add auth data to**: *Request Headers*
   b. In the **Configure New Token** section, click **Edit Token Configuration**.
4. In the boxes that have become available, specify the following values:
   - **Token Name**: `MyStoreIntegration`
   - **Grant Type**: *Password Credentials*
   - **Access Token URL**: `https://localhost/MyStoreInstance/identity/connect/token`
   - **Username**: `admin`
   - **Password**: The password for the `admin` user
   - **Client ID**: The client ID of the application, which you can copy from the **Client ID** box on the Connected Applications (SM303010) form for the MyStoreIntegration client (which you have created in *Example: Registering the Application in Acumatica ERP*)
   - **Client Secret**: The client secret that you have received and saved during the registration of the MyStoreIntegration client on the Connected Applications form
   - **Scope**: `api`
   - **Client Authentication**: *Send client credentials in body*
5. Click **Get New Access Token**. Once the token is received, the **Manage Access Tokens** dialog box opens.

> ⚠ In certain versions of Postman, the approach described in this section does not work. Instead of this approach, you can send a direct `POST` request to the token endpoint. In the body of the request, you should pass the client ID, the client secret, Acumatica ERP username and password, the type of the authorization flow, and the requested scope. For an example of this request, see the Postman `REST.postman_collection.json` collection provided with this course. For details about the parameters passed in the request body, see *Resource Owner Password Credentials Flow* in the documentation.

6. In the **Manage Access Tokens** dialog box, click **Use Token**.
7. On the toolbar of the collection properties window, click **Save**.

Related Links

- *Resource Owner Password Credentials Flow*

Lesson Summary

In this lesson, you have learned how to configure an integration application to use OAuth 2.0 for authorization in Acumatica ERP. You have connected to the token endpoint, passed the client ID and client secret in the authorization header, and requested access to the web service APIs. You have received the access token from Acumatica ERP. You will use this token in subsequent requests to Acumatica ERP.

Lesson 1.3: Signing Out from Acumatica ERP

In this lesson, you will add to the MyStoreIntegration REST application the request that signs out from Acumatica ERP.

If you have granted only the `api` scope to the application, the access token of the application expires in one hour and the session that was opened for this access token is closed automatically. However, if the application has been granted only the `api` scope, we recommend that you call the sign-out method after you have finished your work with Acumatica ERP, because the Acumatica ERP license includes a limit for the number of API users. If you have not signed out, you may have issues with subsequent authorization requests or sign-ins through the API. For details about how to deal with the issues related to the limit for the number of API users during the authorization requests, see *Appendix: Troubleshooting*.

However, if you authorize your integration application to work with Acumatica ERP through OAuth 2.0, the sign-out is not always required after you have finished your work with Acumatica ERP. (This is opposed to the situation when your integration application uses the API methods for the sign-in in Acumatica ERP—that is, uses cookies to manage the application sessions. For these applications, the sign-out is required to close the session each time the work with Acumatica ERP is finished.) For details about when the sign-out is required, see *Additional Information: Session Management in the OAuth 2.0 Applications*.

Lesson Objective

In this lesson, you will learn how to sign out from Acumatica ERP in an OAuth 2.0 application.

Example: Using the REST API

In this example, through the REST API, you will sign out from Acumatica ERP.

To sign out from Acumatica ERP, you will use the `POST` HTTP method and the URL for signing out.

You create a request for signing out from Acumatica ERP in the same way as you did in *Lesson 1.1: Signing In to and Signing Out from Acumatica ERP* in the *I310 Data Retrieval with REST API Basics* training course.

Signing Out from Acumatica ERP

To sign out from Acumatica ERP, do the following:

1. In the Postman collection, add a new request, and configure the settings of the request as follows:
   - HTTP method: `POST`
   - URL: *https://localhost/MyStoreInstance/entity/auth/logout*
   - The headers shown below

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

> ⚠ By default, the request in the collection uses the authorization type specified for the collection —that is, the OAuth 2.0 type, which you have specified in *Example: Configuring the REST Application to Use OAuth 2.0*.

2. Send the request. The response of the successful request includes the `204 No Content` status code.
3. Save the request.

Related Links

- *Sign Out from the Service*

Additional Information: Session Management in the OAuth 2.0 Applications

If you authorize your integration application to work with Acumatica ERP through OAuth 2.0, the sign-out is not always required after you have finished your work with Acumatica ERP.

Whether or not the sign-out is required depends on the access scope that the user has granted to the application as follows:

- If the user has granted only the `api` scope to the application, we recommend that in this case you sign out after you have finished your work with Acumatica ERP. This scenario was described in the example of this lesson.
- If the application has been granted the `api` and `offline_access` scopes (that is, the application has requested a refresh token along with an access token), when the access token has expired, the application can request a new access token by sending a request to the token endpoint and providing the refresh token. Acumatica ERP issues the first access token along with the session ID. If the client application requests a new access token by presenting a refresh token, Acumatica ERP reuses the session ID that was issued for the first access token issued with the refresh token. That is, the system uses a single session for each access granted to the client application. In this case, you do not need to sign out after you have finished your work with Acumatica ERP. This scenario is outside of the scope of this course.
- If the application has been granted the `api:concurrent_access` scope, Acumatica ERP can maintain multiple sessions for the application, managing session IDs through cookies. In this case, the application has to explicitly sign out from Acumatica ERP in each session to close the session. This scenario is outside of the scope of this course.

For details on the scopes that are available for each of the OAuth 2.0 flows, see the descriptions of the flows in the documentation (*Authorization Code Flow*, *Implicit Flow*, and *Resource Owner Password Credentials Flow*).

Lesson Summary

In this lesson, you have learned how to sign out from Acumatica ERP in an OAuth 2.0 application. You have also reviewed whether the sign-out is necessary for the OAuth 2.0 applications.

Part 2: Performance Optimization

In this part of the course, you will learn how to optimize the performance of an application that uses Acumatica ERP integration interfaces. You will perform the minimum number of requests that retrieve all necessary detail lines and find out how to deal with errors that may occur during the retrieval of the list of records through the contract-based REST API. You will also retrieve the records in batches to optimize the performance of the application.

As a result of completing the lessons of this part, you will have experience in optimizing performance of an application that retrieves a list of sales orders with detail lines from Acumatica ERP and a list of sales orders divided into batches. You will also retrieve the list of payment records of a customer one by one, and will learn when you may need to use this approach.

Lesson 2.1: Retrieving a List of Sales Orders with Details and Related Shipments

The online store of the MyStore company should display to a customer the list of sales orders of the customer with details and related shipments. The sales orders are created and maintained on the *Sales Orders* (SO301000) form in Acumatica ERP. That is, the MyStoreIntegration application should be able to export the list of sales orders with multiple kinds of details from Acumatica ERP.

In this lesson, you will retrieve the list of sales orders of the *C00000003* customer. For each sales order in the list, you need to retrieve the following values:

- The order type (**Order Type** of the Summary area of the Sales Orders form)
- The order number (**Order Nbr.** of the Summary area)
- The customer ID (**Customer** of the Summary area)
- The customer order number (**Customer Order Nbr.** of the Summary area)
- The date when the sales order was created (**Date** of the Summary area)
- The ordered quantity (**Ordered Qty.** of the Summary area)
- The order total (**Order Total** of the Summary area)
- For each inventory item in the order (the **Details** tab):
  - The inventory ID (the **Inventory ID** column)
  - The quantity (the **Quantity** column)
  - The unit price (the **Unit Price** column)
- For each shipment in the order (the **Shipments** tab):
  - Shipment number (the **Document Nbr.** column)
  - Invoice number (the **Invoice Nbr.** column)

These elements are shown in the following screenshots.

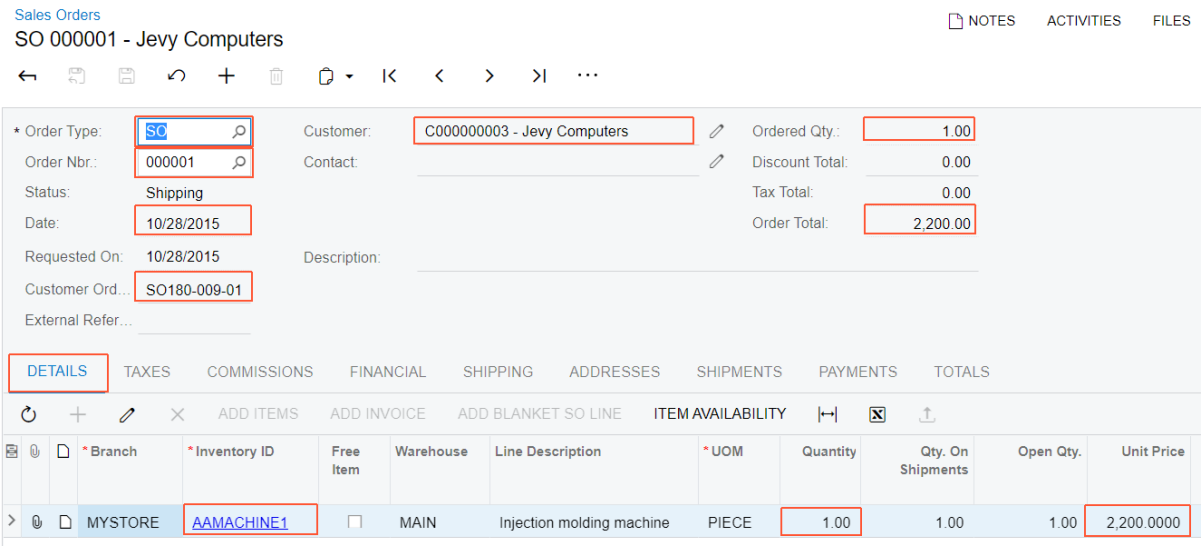

Figure: The Summary area and the Document Details tab

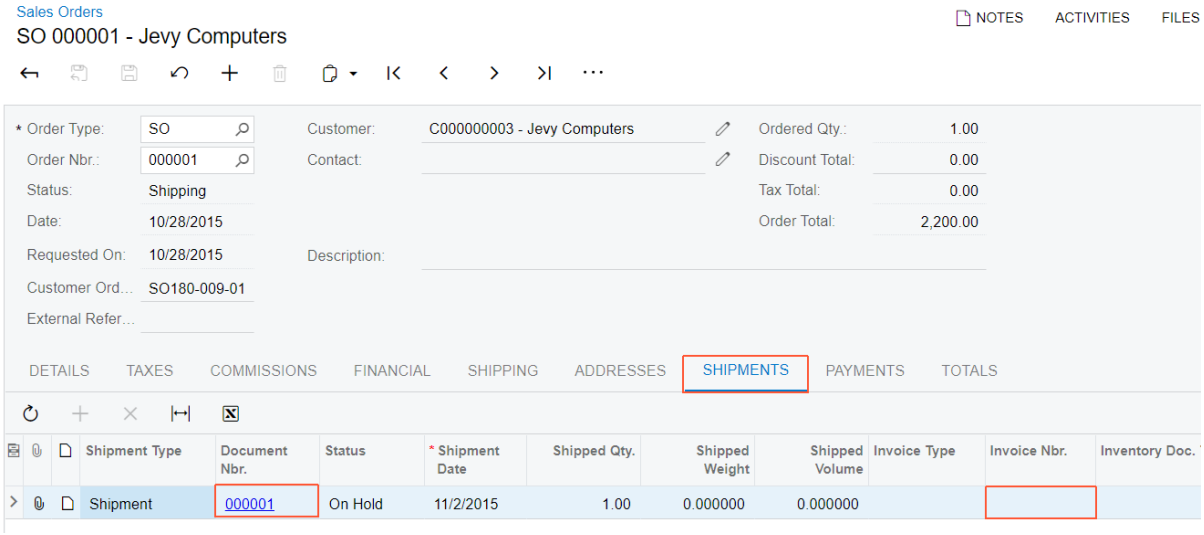

Figure: The Shipments tab

You will use the `SalesOrder` entity of the *Default/20.200.001* endpoint to list the sales orders. The `SalesOrder` entity is mapped to the Sales Orders form. For the best performance of the application, it is important that you request only the values of the fields that you need (instead of requesting the values of all fields available in the entity).

To achieve the best performance of the applications that work with Acumatica ERP through the contract-based REST API, we recommend that you retrieve multiple records with multiple kinds of detail lines in one request.

This lesson shows how you can implement this scenario by using the contract-based REST API. You can also implement this scenario by using the OData interface, but this implementation is outside of the scope of this course. For short information about this implementation, see *Additional Information: Retrieval of the List of Sales Orders Through OData*.

Lesson Objective

In this lesson, you will learn how to retrieve from Acumatica ERP records with multiple kinds of details.

Example: Using One GET Request

In this example, through the REST API, you will configure one HTTP request that exports the list of sales orders with multiple kinds of details.

You will use the `GET` methods with the `$filter`, `$expand`, and `$select` parameters.

Retrieving the List of Sales Orders

To retrieve the list of sales orders with details and related shipments, do the following:

Because you have not requested the refresh token during the OAuth 2.0 authorization of the application, the access token, which you have received during authorization, expires in one hour and you need to request a new access token once the token has expired. In Postman, to receive a new access token, in the request that you configure in the collection, you can do the following:

1. Click the **Authorization** tab and click **Get New Access Token**.
2. In the **Manage Access Tokens** dialog box, click **Use Token**.

1. To retrieve the list of sales orders with summary information and document details, in Postman, configure the following settings of a contract-based REST API request:
   - HTTP method: `GET`
   - URL: *https://localhost/MyStoreInstance/entity/Default/20.200.001/SalesOrder*
   - The following parameters of the request

Parameter	Value
`$filter`	`CustomerID eq 'C000000003'`
`$expand`	`Details,Shipments`
`$select`	`OrderNbr,OrderType,CustomerID,CustomerOrder, Details/InventoryID,Details/OrderQty,Details/UnitPrice,Date, OrderedQty,OrderTotal,Shipments/InvoiceNbr,Shipments/ShipmentNbr`

   - The headers shown below

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

2. Send the request. The response contains the `200 OK` status code. For each sales order of the *C00000003* customer, the body of the response includes the values of the `OrderType`, `OrderNbr`, `CustomerID`, `CustomerOrder`, `Date`, `OrderedQty`, and `OrderTotal` fields, the list of details with the values of `InventoryID`, `OrderQty`, and `UnitPrice`, and the list of shipments with the values of `InvoiceNbr` and `ShipmentNbr`. The following code shows a fragment of the response body.

```
[
```

```
{
    "id": "c01087ca-7281-e511-80c0-00155d012302",
    "rowNumber": 1,
    "note": {
        "value": ""
    },
    "CustomerID": {
        "value": "C000000003"
    },
    "CustomerOrder": {
        "value": "SO180-009-01"
    },
    "Date": {
        "value": "2015-10-28T00:00:00+03:00"
    },
    "Details": [
        {
            "id": "a7d978fb-7281-e511-80c0-00155d012302",
            "rowNumber": 1,
            "note": {
                "value": ""
            },
            "InventoryID": {
                "value": "AAMACHINE1"
            },
            "OrderQty": {
                "value": 1.000000
            },
            "UnitPrice": {
                "value": 2200
            },
            "custom": {}
        }
    ],
    "OrderedQty": {
        "value": 1
    },
    "OrderNbr": {
        "value": "000001"
    },
    "OrderTotal": {
        "value": 2200
    },
    "OrderType": {
        "value": "SO"
    },
    "Shipments": [
        {
            "id": "a726a651-978a-4e38-a6ab-79af69a29007",
            "rowNumber": 1,
            "note": {
                "value": null
            },
            "InvoiceNbr": {},
            "ShipmentNbr": {
                "value": "000001"
            },
```

```
                "custom": {}
            }
        ],
        "custom": {}
    },
    ...
]
```

3. Save the request.

Related Links

- *Retrieve Records by Conditions*
- *Parameters for Retrieving Records*

Additional Information: Retrieval of the List of Sales Orders Through OData

You can implement the integration scenario described in this lesson by using the OData Version 3.0 or Version 4.0 interface.

To export records with multiple kinds of detail lines from Acumatica ERP by using the OData Version 3.0 interface, you need to configure multiple generic inquiries on the *Generic Inquiry* (SM208000) form that export all necessary data (one generic inquiry for each kind of detail lines that you need to export), expose these generic inquiries via OData (by clicking the **Expose via OData** check box on the form), and execute the OData requests. For the best performance of your application, we recommend that you create a separate generic inquiry for each kind of detail lines.

For details about the configuration of generic inquiries, see *Creating a Generic Inquiry* in the documentation, or refer to the *S130 Reporting: Data Retrieval and Analysis* training course.

To export records with multiple kinds of detail lines from Acumatica ERP by using the OData Version 4.0 interface, you use multiple navigation properties of the `PX.Objects.SO.SOOrder` data access class.

For details about the creation and execution of OData requests, see *Exposing an Inquiry by Using OData* in the documentation or refer to the *I300 Data Retrieval with OData* training course.

Lesson Summary

In this lesson, you have added to the MyStoreIntegration REST application the requests that retrieve the list of sales orders with details and related shipments from Acumatica ERP. You have used one request to retrieve two kinds of details, which optimizes the performance of the application. You have also reviewed how this scenario can be implemented with the OData interface.

The following table summarizes the availability of the performance optimization options for different integration interfaces.

Integration Interface	Optimized Retrieval of Multiple Kinds of Detail Lines
OData Version 3.0 interface	Yes, if you create multiple custom generic inquiries (one inquiry for each kind of detail line)
OData Version 4.0 interface	Yes
REST API	Yes

Lesson 2.2: Retrieving a List of Sales Orders in Batches

If a customer of the online store of the MyStore company has a large number of sales orders and these sales order should be displayed to the customer using the online store, the online store can display these sales orders in batches of multiple records for better performance. That is, the MyStoreIntegration application should be able to export the list of sales orders from Acumatica ERP in batches of multiple records.

In this lesson, you will modify the examples of *Lesson 2.1: Retrieving a List of Sales Orders with Details and Related Shipments* to retrieve the customer's sales orders in batches of five records.

This lesson shows how you can implement this scenario by using the contract-based REST API.

You can also implement this scenario by using the OData interface, but this implementation is outside of the scope of this course. For brief information about this implementation, see *Additional Information: Retrieval of the List of Sales Orders in Batches Through OData*.

Lesson Objective

In this lesson, you will learn how to retrieve records in batches from Acumatica ERP.

Example: Using $top and $skip

In this example, through the REST API, you will configure HTTP requests that export the list of sales orders in batches of five records.

You will use `GET` requests with the `$top` and `$skip` parameters. In the first request, you will specify *5* as the value of the `$top` parameter, which means that the top five sales order records should be retrieved from the database. (The records are retrieved from the database sorted by the values of key fields.) In the second request, you will specify *5* as the value of the `$top` parameter and *5* as the value of the `$skip` parameter, which means that the first five sales order records will be skipped and the sixth to tenth records will be retrieved. Subsequent requests can proceed similarly to the second one, with the previously exported records skipped and the next five retrieved.

You will also use the `$filter`, `$expand`, and `$select` parameters.

Retrieving the List of Sales Orders in Batches

To retrieve the list of sales orders in batches, do the following:

1. To retrieve the first five sales orders, in Postman, specify the following settings of a contract-based REST API request:
   - HTTP method: `GET`
   - URL: *https://localhost/MyStoreInstance/entity/Default/20.200.001/SalesOrder*
   - The following parameters of the request

Parameter	Value
`$filter`	`CustomerID eq 'C000000003'`
`$select`	`OrderNbr,OrderType,CustomerID,OrderTotal`

Parameter	Value
`$top`	`5`

- The headers shown below

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

Instead of creating a new request, you can duplicate the request from *Example: Using One GET Request* and adjust its settings.

2. Send the request. The response contains the `200 OK` status code. The body of the response includes five sales order records of the *C00000003* customer with the *000001*, *000004*, *000005*, *000006*, and *000007* order numbers. The following code shows a fragment of the response body.

```
[
    {
        "id": "c01087ca-7281-e511-80c0-00155d012302",
        "rowNumber": 1,
        "note": {
            "value": ""
        },
        "CustomerID": {
            "value": "C000000003"
        },
        "OrderNbr": {
            "value": "000001"
        },
        "OrderTotal": {
            "value": 2200.0000
        },
        "OrderType": {
            "value": "SO"
        },
        "custom": {}
    },
    {
        ...
    },
    {
        ...
    },
    {
        ...
    },
    {
        "id": "1a37a70e-7481-e511-80c0-00155d012302",
        "rowNumber": 5,
        "note": {
            "value": ""
```

```
        },
        "CustomerID": {
            "value": "C000000003"
        },
        "OrderNbr": {
            "value": "000007"
        },
        "OrderTotal": {
            "value": 170.0000
        },
        "OrderType": {
            "value": "SO"
        },
        "custom": {}
    }
]
```

3. Save the request.
4. Add the `$skip` parameter of the request with a value of `5`.
5. Send the request. The response contains the `200 OK` status code. The body of the response includes the remaining four sales order records of the *C00000003* customer with the *000008*, *000009*, *000010*, and *000011* order numbers. The following code shows a fragment of the response body.

```
[
    {
        "id": "c68dfa2f-7481-e511-80c0-00155d012302",
        "rowNumber": 1,
        "note": {
            "value": ""
        },
        "CustomerID": {
            "value": "C000000003"
        },
        "OrderNbr": {
            "value": "000008"
        },
        "OrderTotal": {
            "value": 535.0000
        },
        "OrderType": {
            "value": "SO"
        },
        "custom": {}
    },
    {
        ...
    },
    {
        ...
    },
    {
        "id": "33f41ca8-7481-e511-80c0-00155d012302",
        "rowNumber": 4,
        "note": {
            "value": ""
        },
        "CustomerID": {
```

```
            "value": "C000000003"
        },
        "OrderNbr": {
            "value": "000011"
        },
        "OrderTotal": {
            "value": 170.0000
        },
        "OrderType": {
            "value": "SO"
        },
        "custom": {}
    }
]
```

6. Save the request.

Related Links

- *Parameters for Retrieving Records*

Additional Information: Retrieval of the List of Sales Orders in Batches Through OData

You can implement the integration scenario described in this lesson by using the OData Version 3.0 or Version 4.0 interface.

To export records in batches from Acumatica ERP by using the OData Version 3.0 or Version 4.0 interface, you need to use the `$top` and `$skip` parameters of the request.

For details about the execution of OData Version 3.0 requests, see *Exposing an Inquiry by Using OData*. For details about the creation and execution of OData Version 3.0 and Version 4.0 requests, refer to the *I300 Data Retrieval with OData* training course.

Lesson Summary

In this lesson, you have added to the MyStoreIntegration application the request that retrieves the list of sales orders from Acumatica ERP in batches of five records. You have implemented this scenario by using the contract-based REST API. You have used the `$top` and `$skip` parameters of the REST request. You have also reviewed how this scenario can be implemented with the OData interface.

Lesson 2.3: Retrieving the List of Payments One by One

The online store of the MyStore company should display to a customer the list of payments of the customer. The payments are created on the *Payments and Applications* (AR302000) form in Acumatica ERP. That is, the MyStoreIntegration application should be able to export the list of payments from Acumatica ERP.

This lesson shows how to export payments with details if it is impossible to request the list of payments with details in one request.

In this lesson, you will retrieve the list of payments of the *C00000003* customer. For each payment in the list, you need to retrieve the following values (with the corresponding locations on the Payments and Applications form):

- The reference number (the **Reference Nbr.** box in the Summary area)
- The type (the **Type** box in the Summary area)
- The status (the **Status** box in the Summary area)
- The application date (the **Application Date** box in the Summary area)
- The type and reference number of the document to which the payment is applied (the **Doc. Type** and **Reference Nbr.** columns on the **Application History** tab)

These elements correspond to the following fields of the `Payment` entity of the *Default/20.200.001* endpoint:

- `ReferenceNbr`
- `Type`
- `Status`
- `ApplicationDate`
- `DisplayDocType` and `DisplayRefNbr` of the `ApplicationHistory` detail entity

When multiple records are retrieved from Acumatica ERP through an endpoint with Contract Version 4 (which the *Default/20.200.001* endpoint has), the system tries to optimize the retrieval of the records and obtain all needed records in one request to the database (instead of requesting the records one by one). However, in the examples of this lesson, the optimization fails for the fields of the `ApplicationHistory` detail entity, and the system returns an error. To fix this error, you have the following options:

- If you do not need to retrieve the fields of the `ApplicationHistory` detail entity, you can exclude these fields and the `ApplicationHistory` entity from the request.
- If you need to retrieve the fields of the `ApplicationHistory` detail entity, you can retrieve the needed records one by one by the key fields.

In this lesson, you will use the latter option. That is, you will do the following:

1. Retrieve the list of key fields of the payments of the *C00000003* customer
2. Retrieve the payments with the needed details one by one by using the key fields

To achieve the best performance of the retrieval of the list of payments, you can create a custom generic inquiry, add it to a custom endpoint or an endpoint extension, and use this generic inquiry for the data retrieval. This scenario is outside of the scope of this course. For details about retrieval of the data from a generic inquiry, see *Retrieve Data from an Inquiry Form* in the documentation, or see the *I310 Data Retrieval with REST API Basics* training course.

Lesson Objective

In this lesson, you will learn how to deal with the errors that can occur if the performance optimization fails.

Example: Using GET with Key Field Values

In this example, you will do the following:

1. Try to retrieve the list of payments with `ApplicationHistory` details in one request, which fails.
2. Modify the request to retrieve only the key fields of the needed payments.
3. Configure the request that retrieves the payments one by one by the key fields. To retrieve the record by the key fields, you will specify the values of the key fields in the URL of the request.

If you specify the key field values in the `$filter` parameter instead of passing the key fields in the URL of the request, Acumatica ERP treats this request as a request for multiple records and performs additional optimizations, which are not necessary when you request one record. Therefore, we recommend that you specify the key field values in the URL of the request if you want to retrieve one record.

Retrieving the List of Payments

To retrieve the list of payments, do the following:

1. In Postman, configure the following settings of the contract-based REST API request:
   - HTTP method: `GET`
   - URL: *https://localhost/MyStoreInstance/entity/Default/20.200.001/Payment*
   - The following parameters of the request

Parameter	Value
`$filter`	`Type eq 'Payment' and CustomerID eq 'C000000003'`
`$expand`	`ApplicationHistory`
`$select`	`ReferenceNbr,Type,Status,ApplicationHistory/DisplayDocType, ApplicationHistory/DisplayRefNbr,ApplicationDate`

   - The headers shown below

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

2. Send the request. The request fails with the error that is shown in the following screenshot.

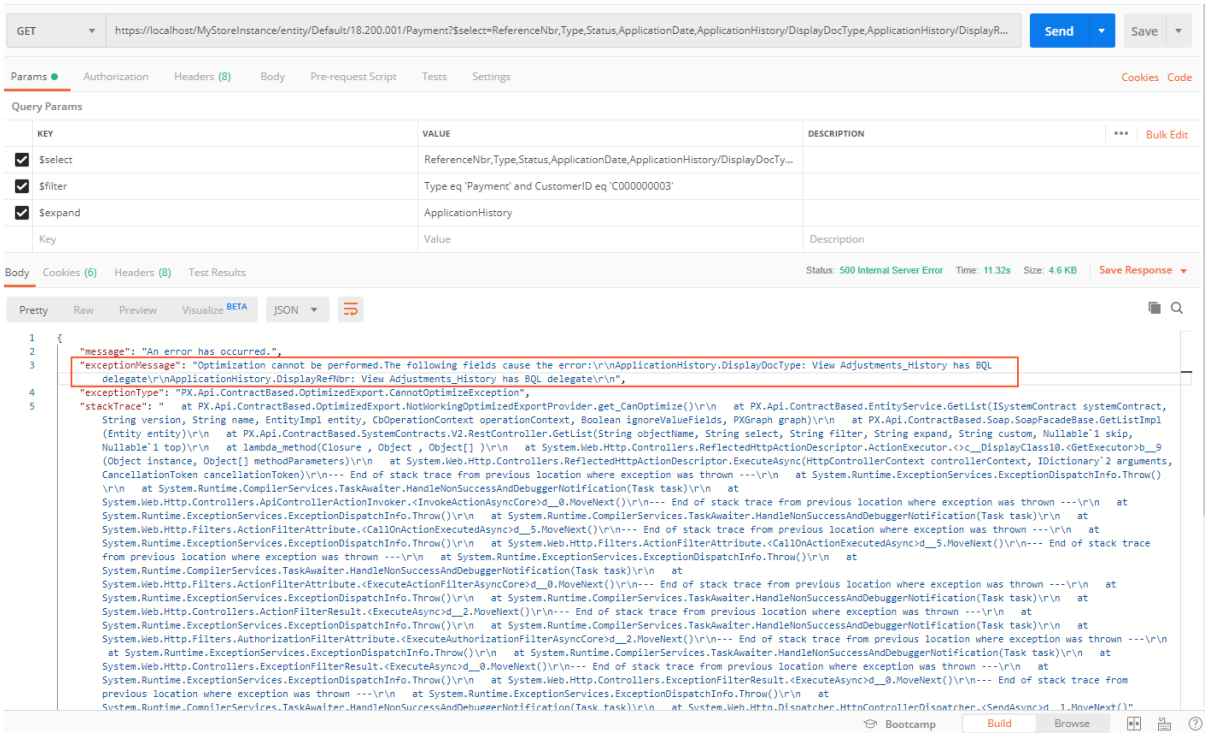

Figure: The returned error

3. Modify the parameters of the request to return only the key fields of the payments as follows.

Parameter	Value
`$filter`	`Type eq 'Payment' and CustomerID eq 'C000000003'`
`$select`	`ReferenceNbr,Type`

4. Send the request. The response contains the `200 OK` status code, and the body includes the values of the `CustomerID`, `ReferenceNbr`, and `Type` fields.
5. Save the request.
6. Configure the request that retrieves the details of the payments by the values of the key fields (`ReferenceNbr` and `Type`) as follows:
   - HTTP method: `GET`
   - URL: *https://localhost/MyStoreInstance/entity/Default/20.200.001/Payment/Payment/000001*
   - The following parameters of the request

Parameter	Value
`$select`	`ReferenceNbr,Type,Status,ApplicationHistory/DisplayDocType, ApplicationHistory/DisplayRefNbr,ApplicationDate`
`$expand`	`ApplicationHistory`

   - The headers shown below

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

7. Send the request. Make sure the HTTP status code is `200 OK`, and review the returned data, of which an example is shown below.


```
{
    "id": "525ec24d-e03b-4628-a430-712163515bc2",
    "rowNumber": 1,
    "note": {
        "value": ""
    },
    "ApplicationDate": {
        "value": "2018-10-12T00:00:00+03:00"
    },
    "ApplicationHistory": [
        {
            "id": "ad31f8d4-04a0-45b8-9cf5-efd1b20cb664",
            "rowNumber": 1,
            "note": {
                "value": ""
            },
            "DisplayDocType": {
                "value": "Invoice"
            },
            "DisplayRefNbr": {
                "value": "INV000045"
            },
            "custom": {}
        }
    ],
    "ReferenceNbr": {
        "value": "000001"
    },
    "Status": {
        "value": "Closed"
    },
    "Type": {
        "value": "Payment"
    },
    "custom": {}
}
```


8. Save the request.
9. Repeat the request for each pair of payment key fields in the list.

If you performed only the examples described in this guide, the *C00000003* customer has only one payment; therefore, you do not need to perform other requests to retrieve the details of all payments of the customer.

Related Links

- *Retrieve Records by Conditions*
- *Retrieve a Record by Key Fields*

Lesson Summary

In this lesson, you have added to the MyStoreIntegration REST application the methods that retrieve the payments of a customer one by one. You have used this approach because the retrieval of the payments with the specified fields could not be optimized for performance.

Part 3: Retrieval of Attachments

In this part of the guide, you will use the contract-based REST API to retrieve the attachments of the stock item records from Acumatica ERP.

Attachments cannot be retrieved from Acumatica ERP through OData requests.

Lesson 3.1: Retrieving the Attachments of a Stock Item

In this lesson, you will add to the MyStoreIntegration application a method that retrieves the files that are attached to a stock item record.

The MyStore company needs to display an image of each item that is sold in the online store. Images for the items can be stored in Acumatica ERP as attachments to the *Stock Items* (IN202500) form. To display an image of a stock item in the online store, the MyStoreIntegration application should export the images that are attached to the stock item. The name of the image is not known when the online store requests the file from Acumatica ERP.

Lesson Objective

In this lesson, you will learn how to retrieve the files that are attached to a stock item by using the contract-based REST API.

Prerequisites

On the *Stock Items* (IN202500) form, select the record with the *AAMACHINE1* inventory ID. On the title toolbar, click **Files** and notice that one file with the name `T2MCRO.jpg` is attached to the record, as shown in the following screenshot. In the examples of this lesson, you will export this file from Acumatica ERP.

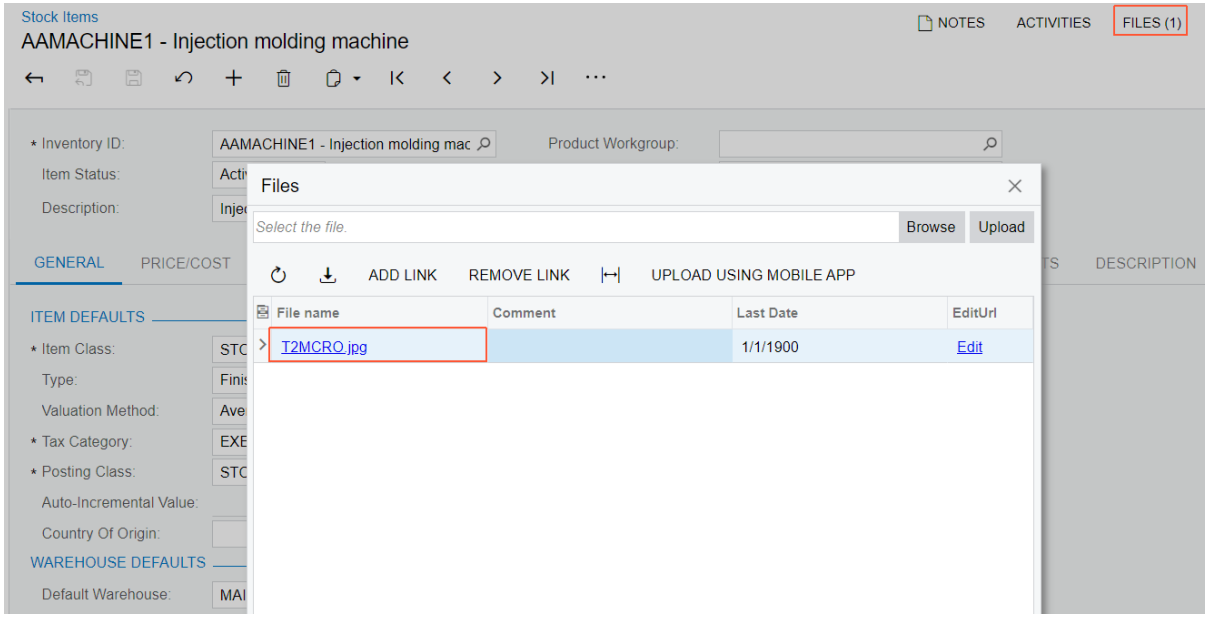

Figure: Attached file

Example: Using the GET Method

In this example, you will configure an HTTP request that exports the files attached to a stock item record by using the contract-based REST API.

You will obtain the needed stock item record by using the values of its key fields. You will specify the key fields in the URL of the `GET` request. You will retrieve only the ID of the stock item and the values of the system fields, which are always returned. Then you will retrieve the attached file by using the `GET` method and the link that will have been returned in the `href` field of an entity in the `files` array.

Retrieving the Attachments of a Stock Item

To retrieve the attachments of a stock item, do the following:

1. In Postman, configure the following settings of the request that retrieves the stock item with the *AAMACHINE1* ID:
   - HTTP method: `GET`
   - URL: *https://localhost/MyStoreInstance/entity/Default/20.200.001/StockItem/AAMACHINE1*
   - Parameters:
     - `$select=InventoryID,files`
     - `$expand=files`
   - The headers shown below

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

2. Send the request. The response of the successful request contains the `200 OK` status code and includes in the body the links to the files attached to the stock item record, as the following code example shows.

```
{
    "id": "2537440a-52c5-4c79-9382-5a78ec581f1e",
    "rowNumber": 1,
    "note": {
        "value": ""
    },
    "InventoryID": {
        "value": "AAMACHINE1"
    },
    "custom": {},
    "files": [
        {
            "id": "9be45eb7-f97d-400b-96a5-1c4cf82faa96",
            "filename": "Stock Items (AAMACHINE1)\\T2MCRO.jpg",
            "href": "/MyStoreInstance/entity/Default/20.200.001/files/9be45eb7-
f97d-400b-96a5-1c4cf82faa96"
        }
    ]
}
```

3. Send the `GET` request to the URL returned for the attached file (*https://localhost/MyStoreInstance/entity/Default/20.200.001/files/9be45eb7-f97d-400b-96a5-1c4cf82faa96*). The response of a successful request contains the file in the response body.

Related Links

- *Retrieve a Record by Key Fields*
- *Retrieve a File Attached to a Record*

Lesson Summary

In this lesson, you have learned how to export the files that are attached to a record.

The following table summarizes whether the attachment can be retrieved from Acumatica ERP through the use of the integration interfaces.

Integration Interface	Retrieval of Attachments
OData Version 3.0 interface	No
OData Version 4.0 interface	No
REST API	Yes

Appendix: Comparison of the Integration Interfaces

The following tables summarize the differences between the integration interfaces described in this training guide. For each integration interface, the tables show whether the scenario mentioned in the header cell can be implemented with this integration interface.

OAuth 2.0 Authorization

Integration Interface	OAuth 2.0 Authorization
OData Version 3.0 interface	Yes
OData Version 4.0 interface	Yes
REST API	Yes

Optimized Retrieval of Multiple Kinds of Detail Lines

Integration Interface	Optimized Retrieval of Multiple Kinds of Detail Lines
OData Version 3.0 interface	Yes, if you create multiple custom generic inquiries (one inquiry for each kind of detail line)
OData Version 4.0 interface	Yes
REST API	Yes

Retrieval of Attachments

Integration Interface	Retrieval of Attachments
OData Version 3.0 interface	No
OData Version 4.0 interface	No
REST API	Yes

Appendix: Web Integration Scenario Reference

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

- **Authorizing the application to work with Acumatica ERP**: *Lesson 1.1: Registering the Application in Acumatica ERP*, *Lesson 1.2: Configuring the Application to Use OAuth 2.0*
- **Signing out from Acumatica ERP in an OAuth 2.0 application**: *Lesson 1.3: Signing Out from Acumatica ERP*
- **Listing the sales orders of a customer**: *Lesson 2.1: Retrieving a List of Sales Orders with Details and Related Shipments*
- **Listing the payments of a customer**: *Lesson 2.3: Retrieving the List of Payments One by One*
- **Retrieving the images of a stock item**: *Lesson 3.1: Retrieving the Attachments of a Stock Item*

Appendix: Troubleshooting

I get the error *API Login Limit* when my application requests access to the Acumatica ERP web services API through OAuth 2.0. What should I do?

For an application that uses OAuth 2.0 for authorization in Acumatica ERP, this error appears if all of the following are true:

- The API login limit is specified in the Acumatica 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 Monitoring Console* (SM604000) form.
- The number of unclosed sessions (that is, the sessions in which you have signed in to Acumatica ERP through the web services API or obtained access to the Acumatica ERP web services API through OAuth 2.0 and have not signed out from Acumatica ERP) equals the API login limit in the license.
- You try to request access to the web services API through OAuth 2.0 once more.

You can deal with this error as follows:

1. Modify the code of your application so that it signs out from Acumatica ERP each time the work with Acumatica ERP is finished. For details, see *Lesson 1.3: Signing Out from Acumatica ERP*.
2. If the integration application has not closed the session, you can do one of the following:
   - Pass the access token that was used during the previous session and sign out from Acumatica ERP.
   - Wait for one hour until the session that has been opened through OAuth 2.0 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.

When I retrieve the list of records from Acumatica ERP through the contract-based REST API, I get the error *Optimization cannot be performed. The following fields cause the error*. What should I do?

When multiple records are retrieved from Acumatica ERP through an endpoint with Contract Version 4, the system tries to optimize the retrieval of the records and obtain all needed records in one request to the database (instead of requesting the records one by one). If the optimization fails, the system returns an error, which specifies the entities or fields that caused the failure of the optimized request. To prevent the error from occurring, you can do one of the following:

- If you do not need to retrieve the entities or fields that caused the failure, you can exclude these entities or fields from the REST API request as follows:
  - Exclude the entities from the entities specified in the `$expand` parameter.
  - Explicitly specify the other fields to be returned (while excluding the fields that caused the failure) by using the `$select` parameter.
- If you need to retrieve the entities or fields that caused the failure, you can retrieve the needed records one by one either by key fields or by IDs.

For an example of how to deal with the optimization errors, see *Lesson 2.3: Retrieving the List of Payments One by One*.