Integrating with Demandware

Student Guide

demandware
A Salesforce Company

About the Course

Audience

Developers

Duration

6 Hours

Course Materials

- Integrating with Demandware Student Guide
- Demandware product documentation

Note: If you need assistance, email education.instructor@demandware.com

Course Prerequisites:

- Complete the Developing in Demandware certification.
- Hands-on experience developing in Demandware

demandware
A Salesforce Company

This course is aimed at developers who have completed the Developing in Demandware course and its associated certification. It should take you approximately 6 hours to complete if you perform all of the hands on activities and use all of the resources associated with the course. To do the activities, you will need UX-Studio connected to your sandbox.

You can find the Student Guide on the Course Materials tab. You should also have access to the Demandware product documentation.

About the Course

Course Icons:

Other Considerations

No Audio

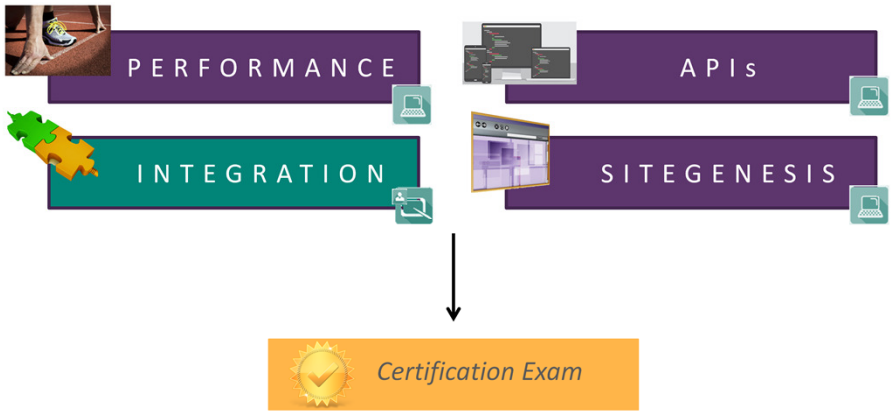
Demandware Best Practice

Use Cases or Case Study

Don't Forget

You may see the following icons as you progress through this course.

Curriculum



Certification, Integration, and Performance contains four courses, which you can take in any order. Upon completion of all four, you can take the certification exam. We recommend that you complete all of the reading and activities prior to scheduling your exam.

Course Contents

01 Integrating with Demandware Overview

- Typical integrations
- Planning for integration
- Integration protocols
- Designing for integration

02 Transferring Data Asynchronously

- Using WebDAV
- Scheduling and monitoring jobs using the Integration Framework

03 Using Web Services for Synchronous Integration

- Integration methods
- Using REST for integration
- Using the web service framework

04 Integrating Using the Open Commerce APIs

- Configuring the Open Commerce API
- Using the Open Commerce API

demandware
A Salesforce Company

This course provides an overview of integration with Demandware and then goes into more detail about asynchronous and synchronous integration.

Integrating with Demandware

Course Learning Objectives

Upon completion of this course you will be able to:

- Differentiate between the various integration types and their usage in Demandware.
- Use the Integration Framework to schedule and monitor jobs when importing asynchronously to Demandware.
- Leverage web services to integrate external information into a site.
- Use the Open Commerce APIs to extend the functionality of a site.

Upon completion of this course, you will be able to:
Differentiate between the various integration types and their usage in Demandware.
Use the Integration Framework to schedule and monitor jobs when importing asynchronously to Demandware.
Leverage web services to integrate external information into a site.
Use the Open Commerce APIs to extend the functionality of a site.

demandware
A Salesforce Company

01 Integrating with Demandware Overview

Module Objectives

Upon completion of this module you will be able to:

- Differentiate between the Demandware integration types.
- Identify use cases for each integration type.
- Identify integration planning considerations.

Upon completion of this module you will be able to:
Differentiate between the Demandware integration types.
Identify use cases for each integration type.
Identify integration planning considerations.

Introduction: Typical Integrations

Before performing the actual tasks involved in setting up and executing any integration, it is important to plan for integration as part of your overall site design. It is important to understand the big picture for your current and future needs in order to ensure that the site can scale in the future as well as take advantage of any platform functionality or third party systems that your organization plans to use.

eCommerce sites generally leverage integration with external systems to provide customer information, payment information, or product information. However, the shopping experience must be seamless to the customer.

Typical integrations include:
ERP / Order Management
Payment
Fraud check/authorization
Customer Relationship Management (CRM)
Personal Information Management (PIM)
Rich Media

Planning Considerations

- How do integrations fit into the business processes?
- What are the use cases?
- What are the sending and receiving systems as well as their dependencies?
- What data needs to be transmitted?
- What are the characteristics of that data (such as: volume, frequency, change, and growth)?
- Is real-time necessary or can it be batched?
- Are there security considerations?

While this is not an exhaustive list, here are some questions to consider:
How do integrations fit into the business processes?
What are the use cases?
What are the sending and receiving systems as well as their dependencies?
What data needs to be transmitted?
What are the characteristics of that data (such as: volume, frequency, change, and growth)?
Is real-time necessary or can it be batched?
Are there security considerations?

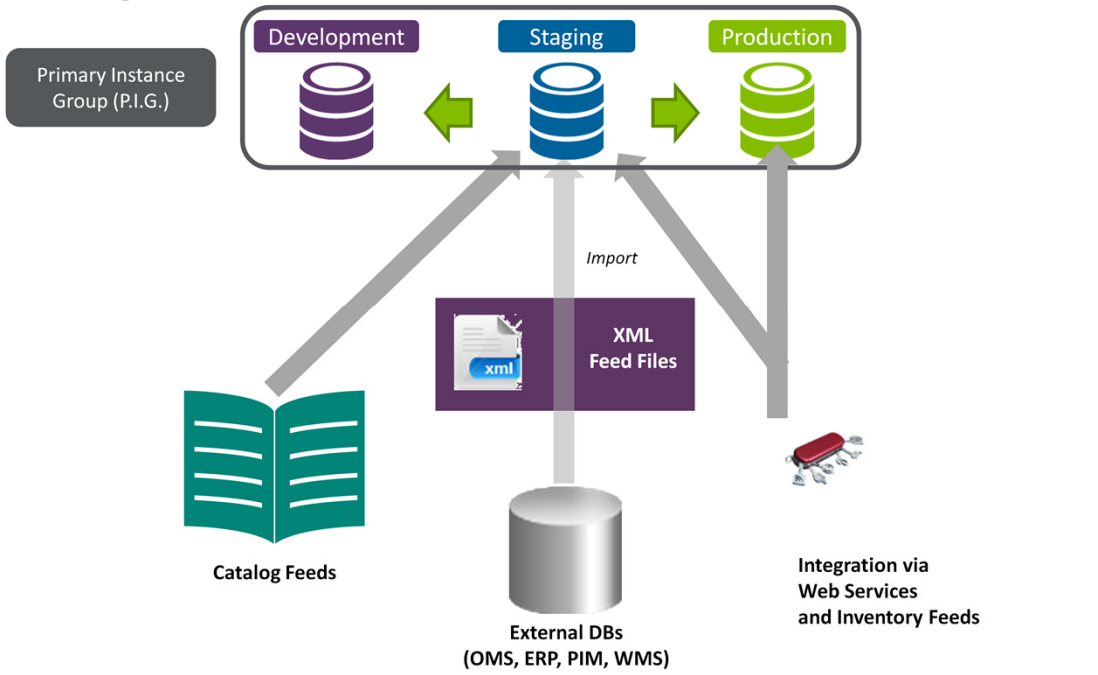
Just Thinking

What are some ways that your organization plans to use integration in your site?

What are some of the key considerations for these decisions?

Jot down some ideas.

Integration Protocols



The Demandware platform provides mechanisms for integrating with third party systems. The platform's service-oriented architecture supports the use of several data transfer protocols. These protocols allow the platform to send and receive data both synchronously and asynchronously. For some data, it is critical to deliver it in real-time, some data can be delivered in near-real-time, and some can be batched.

Asynchronous Data Transfer

Protocol	Description
FTP/sFTP	Supports upload and download of files such as XML and CSV, with Demandware acting as an FTP client for text or binary file transfer. You can also create an FTP client within your instance using the Demandware script class **FTPClient**.  **Note**: For more information on FTP, see the Demandware documentation at `https://info.demandware.com`
WebDAV	Manages files and folders on Demandware servers.

Asynchronous data transfers enable you to import data to the Demandware platform. Because they are file-based, they can accommodate the transfer of large amounts of data. Some examples of asynchronous data transfers include imports of: catalog, product, pricing, and inventory information. Additionally, you can export data from the Demandware platform, such as an XML sitemap to external search engines.

While asynchronous data transfers provide the advantage of transferring large amounts of data quickly, their disadvantage is their high use of system resources.

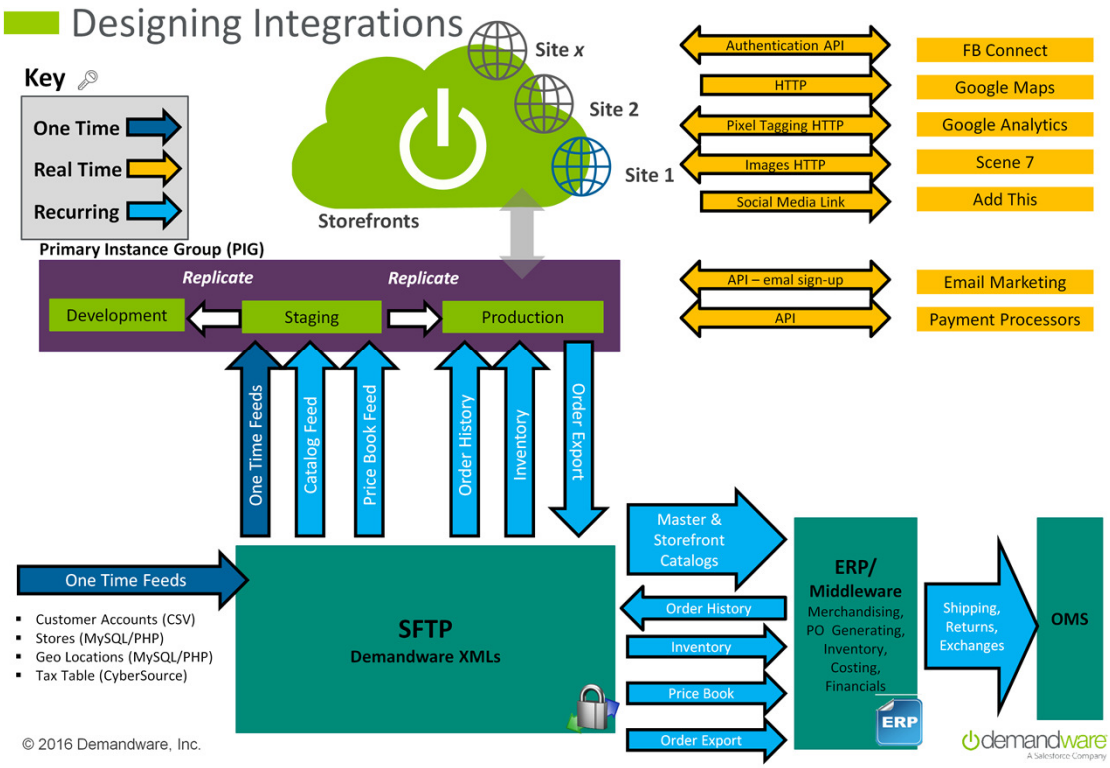
In addition, you can schedule batch jobs to occur once or according to a schedule using the Demandware Integration Framework. It provides a way to schedule, manage, and monitor jobs.

Synchronous Data Transfer

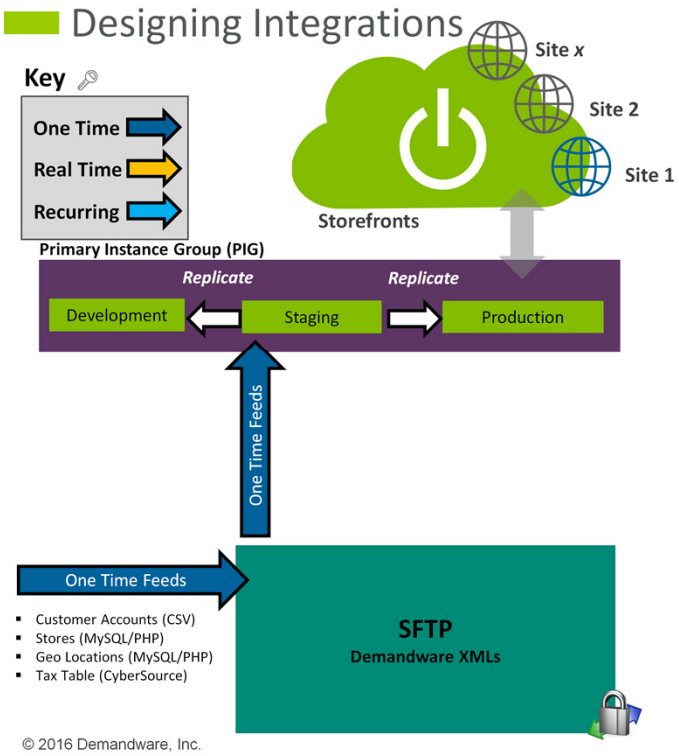
Approach	Definition
SOAP	Simple Object Access Protocol (SOAP) is an XML-based, extensible message envelope format, with 'bindings' to underlying protocols (such as HTTP, SMTP and XMPP). The WSDL (Web Services Definition Language) provides an extensible message envelope that provides the mechanism to implement these bindings.
REST	Representation State Transfer (REST) is a set of architectural principles by which you can design web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages.

Synchronous data transfers are also used to transfer data to, or from, the Demandware platform. With real-time services, the Demandware server invokes a service via a pipeline, SOAP, HTTP Client, or Open Commerce API and then waits for a response to the request. Real-time services exchange data rather than files, thus they typically transfer much smaller amounts of information. As a result, they are less resource intensive than asynchronous data transfers. Demandware supports many types of URLs and can transfer various content items.

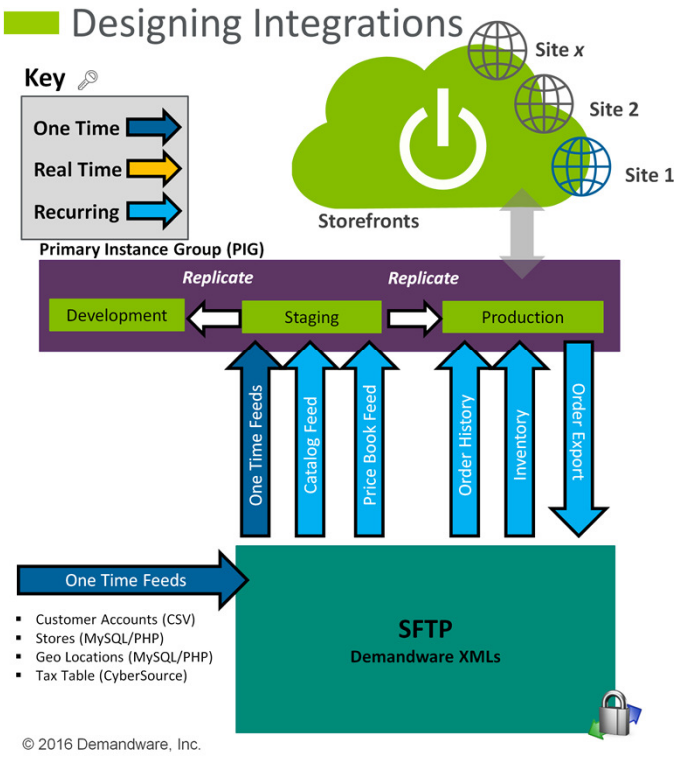
Each has benefits and challenges.



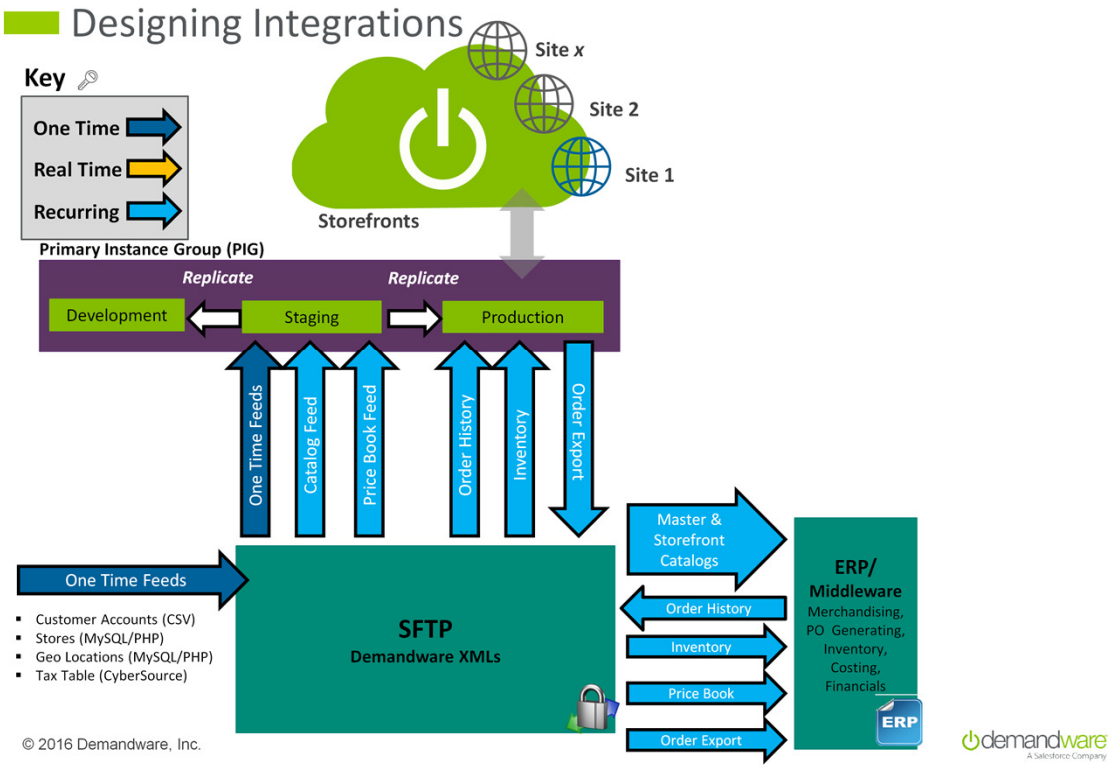
Here is a sample that includes real-time synchronous integrations, recurring asynchronous integrations, and one-time asynchronous integrations.



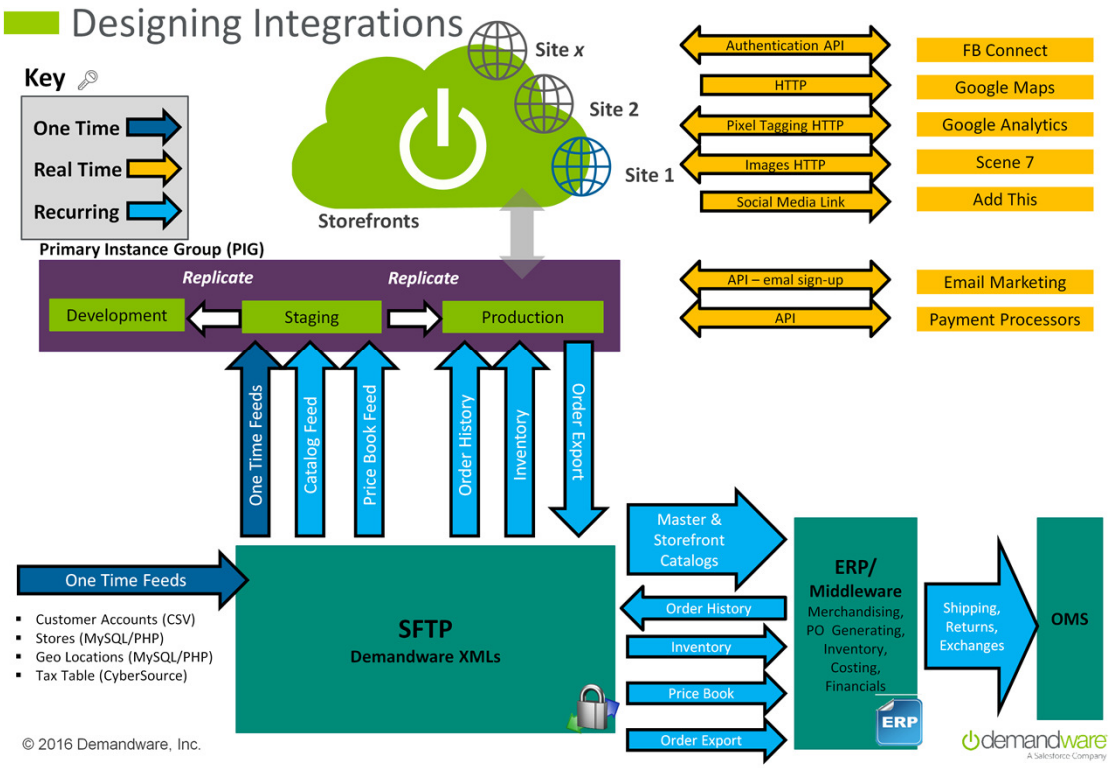
One time feeds are useful for large files because SFTP has less limitations relative to file size. For example, tax tables have a large amount of data, but they do not change often.



Other feeds may be recurring, such as inventory. And, some feeds may be either one time or recurring, depending on your business needs. For example, some organizations' price books change more frequently than others. Notice that the feed could go in either direction, based on what is appropriate for the business.



Sometimes you want to pass the information using middleware, such as an ERP system. Again, notice that the feed could go in either direction, based on what is appropriate for the business. You could export orders, or you might import customer order histories.



That middleware, in turn could pass information to your Order Management System (OMS). Or, you could pass data to your OMS directly.

Just Thinking

Which protocols will your organization most likely use?

Jot down some ideas.

Draw a diagram similar to the one shown in the previous example to represent your system integration.

Now that you know about the types of integration that Demandware supports, let's look at asynchronous integration in more detail.

02 Transferring Data Asynchronously

Module Objectives

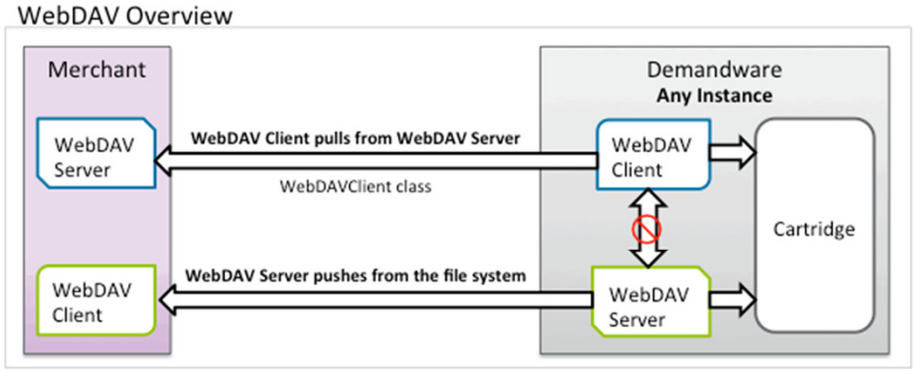
Upon completion of this module you will be able to:

- Describe the purpose and usage of WebDAV in the Demandware platform.
- Use WebDAV to perform asynchronous data transfer.
- Explain the function of the Integration Framework.
- Define the Integration Framework Workflow.
- Create and schedule a batch job.
- Monitor the progress of a batch job.

After completing this module, you will be able to:
Describe the purpose and usage of WebDAV in the Demandware platform.
Use WebDAV to perform asynchronous data transfer.
Explain the function of the Integration Framework.
Define the Integration Framework Workflow.
Create and schedule a batch job.
Monitor the progress of a batch job.

Every Demandware instance has an embedded WebDAV Server and a WebDAV Client. Using a third party WebDAV client application, administrators and developers can manage files (upload, download, move, rename, create, and delete) on the Demandware platform. To successfully transfer files to the Demandware platform using a WebDAV client, you must have Business Manager administration privileges.

WebDAV and Demandware



When transmitting files to the Demandware platform, be aware of file size limitations. For specific information, refer to the Demandware documentation. https://info.demandware.com

When transmitting files to the Demandware platform, be aware that file size limitations exist. For specific information regarding file size limitations, refer to the Demandware documentation at https://info.demandware.com

WebDAV Considerations

- Supports both binary and compressed transmission
- Uses HTTP or HTTPS (PCI compliant)
- Transfer to staging and/or production
- Timestamp updated:
  - Each WebDAV transmission
  - Files are renamed, copied, modified, or created via a WebDAV connection.
- Requires authentication when connecting

When transmitting files to the Demandware platform, be aware of file size limitations. For specific information, refer to the Demandware documentation.

demandware
A Salesforce Company

The WebDAV client can use either an HTTP or HTTPS connection. When the WebDAV client uses an HTTPS connection, the connection becomes PCI compliant. Therefore, WebDAV is safe to transfer files to both staging and production servers.
The WebDAV protocol supports the transmission of both binary and compressed files (for example, WinZIP, GZIP). When transferring compressed files that contain a directory, it maintains the compressed directory structure.
When a file is written to the Demandware platform via any WebDAV connection, the WebDAV server resets the file's timestamp to the current date and time. This timestamp rewrite also occurs when files are renamed, copied, modified or created via a WebDAV connection.
Note: When transmitting files to the Demandware platform, be aware that file size limitations exist. For specific information regarding file size limitations, refer to the Demandware documentation at https://info.demandware.com
Demandware requires authentication when making a WebDAV connection to a Primary Instance Group, therefore all files transferred to a sandbox must be manually transferred via a third party WebDAV utility.

Using WebDAV

In this lab you will:

- Download a third party WebDAV client of your choosing (such as Cyberduck) and install it on your machine.
- The Learning Management System contains practice files associated with the course (catalogstoImport.zip). Using the WebDAV client, transfer the files to your sandbox. Use the Demandware documentation if you need assistance.

Data Transfers Using the Integration Framework

Now that you know about asynchronous data transfers, let's look at how to schedule and monitor data transfers through the use of jobs and the Integration Framework. There are three types of job execution that you can perform within Demandware: scheduled, import/export, and batch. You can schedule jobs to run according to a schedule or for one time only. Demandware also provides a means of detecting problems with job execution, notification of job status and recovery from failed jobs. For instance, if a job fails to execute successfully, you can send an email notification to alert someone to the problem. This feature enables job failure rules to be set, and jobs to automatically re-run if execution fails. You can set job failure rules and retry parameters such as number of retries.

You can define jobs that execute on the global (or organization) level and jobs that execute only on a site level through Administration > Operations.

Using the Integration Framework

In this demonstration, you will see how to:

- Use the Integration Framework to schedule and monitor jobs in the Demandware environment.

Play Video

Knowledge Check

Which of these statements about the Integration Framework is True?

Select all that apply.

X a. It is included as native functionality in the Demandware platform.

✓ b. You can create multiple components in each workflow.

✓ c. The Workflow Plan displays the workflow name, configured workflow components and log information.

Next Steps

- Log onto bitbucket.org and create a new user account.
- Contact developer.community@demandware.com to associate your bitbucket user account to with the Community Suite.

Now that you understand the Integration Framework, consider creating your own bitbucket account and gaining access to the Integration Framework area within bitbucket so that you can ensure that you have access to the most recent versions of the cartridges and its associated documentation.
To do this:
Log onto bitbucket.org and create a new user account.
Contact developer.community@demandware.com to associate your bitbucket user account with the Community Suite.

Web Services for Synchronous Integration

Web services provide a standard for applications to leverage real-time communication using HTTP. They communicate by sending requests and responses using a standard format, a common set of verbs, and correct authentication (based on the API's specification). Using HTTP, the client sends a request and waits for a response from the HTTP server before performing other tasks. Upon receiving the request, the HTTP server provides an immediate response (barring unforeseen circumstances). Clients can reside anywhere (for example, desktop, tablet, mobile device, Internet device).

demandware
A Salesforce Company

03 Using Web Services for Synchronous Integration

Module Objectives

Upon completion of this module you will be able to:

- Differentiate between SOAP and REST integration methods supported by Demandware.
- Describe the REST paradigm and how it manifests itself in the Demandware platform.
- Use REST to create a web service.
- Describe the use of the web service framework.

Upon completion of this module you will be able to:

Differentiate between SOAP and REST integration methods supported by Demandware.
Describe the REST paradigm and how it manifests itself in the Demandware platform.
Use REST to create a web service.
Describe the use of the web service framework.

demandware
A Salesforce Company

SOAP vs. REST

SOAP	REST
▪ Standard XML-based messaging format ▪ Uses packets bound via WSDL ▪ Extensive security standards ▪ Extensible ▪ Flexible (HTTP, or FTP/SMTP) ▪ Can be stateful	▪ An architecture, not a messaging format ▪ Better performance ▪ Scalable ▪ Standard, language independent URIs (less metadata; small and fast). ▪ Stateless ▪ Can use JSON

demandware
A Salesforce Company

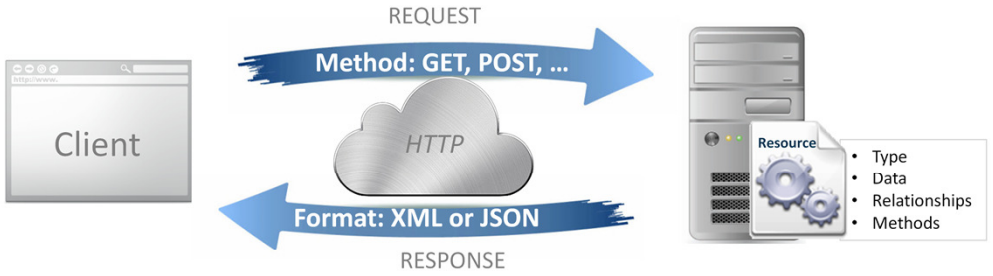
While there are many web service standards, the most commonly used by Demandware developers are SOAP and REST.

SOAP-based services don't deal directly with underlying XML messages; rather, they use XML packets and call functions previously defined in the service that they are calling. The client validates requests before sending. SOAP is known for its extensive standards for security.
SOAP's performance overhead and complexity has resulted in the adoption of web services using JSON in conjunction with REST. RESTful requests are smaller and more concise because they contain less information or metadata than a SOAP request. REST is increasingly popular for mobile development where optimizing size and bandwidth is crucial.
The REST architecture is based on individual HTTP methods (such as GET or POST) to perform specific server-side actions. You send requests via URLs, which typically have parameters. The request format and parameters must match the server-side application specification.
While SOAP uses a standard format, RESTful services are flexible. The data returned can be in any format, but Demandware supports XML and JSON. JSON has less metadata and decorative tags, so that its packets are much smaller than XML. RESTful services can also return non-text values, such as binary files, images, PDF documents, and so on.
Because RESTful services are stateless, each request-response conversation stands on its own. You can cache the information to conserve resources. Being stateless, REST provides improved performance. It does not store client context on the server between requests, therefore each request contains all the information necessary to handle that request. The client stores any all state data (for example, by using cookies).
A REST-based service can use intermediary components to reduce latency, enforce security, and encapsulate legacy systems.

Note: To learn what format a particular service will work with, read the documentation for that service—what kind of requests are accepted, how the data should be passed in and how the data is returned.

Because developers primarily leverage REST with Demandware, this module will focus on REST implementations. For information on SOAP implementation, see the "Using SOAP for Web Service Integrations" white paper in the course materials menu.

REST API



RESTful APIs obtain resources via URLs that make requests based on a limited set of methods (for example, GET, POST). The service determines the syntax and HTTP methods supported. For example, a read-only service might only support GET and POST requests. RESTful APIs use resources which are object instances that describe the available resource type(s), associated application data, behavior and relationships to other resources. Resources can be grouped into collections that contain a single type of resource.

REST handles different media types, based on the supported data (for example, JSON, XML). Currently, the Demandware Web API supports JSON and XML. By default, REST services return JSON if the format is not explicitly specified in the URL.

Specifying the URL

Production

`http(s)://<publicdomain>/dw/<api_type>/`

Example
`http://www.mysite.com/dw/shop/v1/basket/!checkout`

Sandbox

`http(s)://<sub_domain>.demandware.net/s/<site_id>/dw/<api_type>/`

Example
`http://ajones.inside-na02.dw.demandware.net/s/SiteGenesis/dw/shop`

A RESTful API has one entry point, which is an individual resource. Each resource has its own unique URL, which must be communicated to the API clients so that they can find the API.
It is common for the entry point to contain information on the API version, supported features, and resource list. Each collection and resource in the API has its own URL.

Production

http(s)://<publicdomain>/dw/<api_type>/

Example
http://www.mysite.com/dw/shop/v1/basket/!checkout

Sandbox

http(s)://<sub_domain>.demandware.net/s/<site_id>/dw/<api_type>/

Example
http://ajones.inside-na02.dw.demandware.net/s/SiteGenesis/dw/shop

Methods

Method	Description
GET	Retrieve a specific resource(s) from the server.
POST	Create a new resource on the server.
PUT	Update a resource on the server, providing the **entire resource**.
PATCH	Update a resource on the server, providing **only changed attributes**.
DELETE	Remove a resource from the server.
HEAD	Retrieve resource metadata.
OPTIONS	Retrieve what the consumer is allowed to do with the resource.

Methods execute on resources via their URLs. Not all resources support all methods. For detailed information on specific methods, see your Demandware documentation.

URL Example: Get a Specific Product Price

```
REQUEST:
GET /dw/shop/v14_6/products/99225/prices?client_id=14aacef2-9cb7-
4ed4-91d5-04ceb1d3cb8b&locale=en&pretty_print=true
Host: sitegenesis.com
Version: v14_6
Resource Type: Products
resource_id:  /products/{id}/prices
```

Get a specific product price. The product ID on the example shown is: 99225.

REQUEST:
GET
/dw/shop/v14_6/products/99225/prices?client_id=14aacef2-9cb7-4ed4-91d5-04ceb1d3cb8b&locale=en&pretty_print=true
Host: sitegenesis.com
Version: v14_6
Resource Type: Products
resource_id: /products/{id}/prices

URL Example: Get Multiple Promotions by ID

```
REQUEST:
GET /dw/shop/v14_6/promotions/(15off-gps,20off-tvs)
Host: sitegenesis.com
Version: v14_6
Resource Type: Promotions
resource_id:  /products/{id}/prices
```

This example gets multiple promotions by ID.

REQUEST:
GET /dw/shop/v14_6/promotions/(15off-gps,20off-tvs)
Host: sitegenesis.com
Version: v14_6
Resource Type: Promotions
resource_id: /products/{id}/prices

URL Example: Add a Specific Product to Cart on the Mobile App

```
REQUEST:
POST /dw/shop/v13_6/basket/this/add HTTP/1.1
Host: sitegenesis.com
Cookie:
dwsid=tYlzC3YbZNo13dV5XS4OGzg0wClZGz4yThXHrvEZNUlT2ohYzMFyPJin5cW0wleUaxM
nraXcEbg4mnymdroMlA==;
  dwanonymous_9727b83e8e864fa4b6902a37bc70a12d=bcdlZDxB7h5YakHw3p1ZTDPihp
Content-Length: 67
{
  "product_id" : "456",
  "quantity" : 1.00
  "inventory_id" : "0815",
}
Version: v14_6
Resource Type: Basket
resource_id:  /basket/this
```

This example shows how to add a specific product to the cart on the mobile app. Notice the use of the JSON notation.

Create a URL for REST

In this lab you will create and test URLs using a REST utility.

1. Download a third-party tool such as Curl or Postman.
2. Try submitting the two GET requests from the prior example.

Putting it All Together

Next, add the URL to the code which:

- Opens the connection
- Submits the request
- Receives and parses the response.

```
// send the http request to amazon.com
var httpSvc = new HTTPClient();
httpSvc.open( "GET",
          "http://webservices.amazon.com/onca/xml?Service=AWSECommerceService&" +
          "SubscriptionId=YOUR-SUBSCRIPTION-ID&Operation=ItemSearch&" +
          "SearchIndex=Video&Keywords=potter%20harry&ResponseGroup=Images",
          false,
          null,
          null );
httpSvc.send();
// convert the result into an XML object
var result = new XML( httpSvc.text );
// print some values from the result.
// We must use namespace qualified access to read the values.
var ns = new Namespace(
          "http://webservices.amazon.com/AWSECommerceService/2005-07-26" );
trace( result.ns::Items.ns::Request.ns::IsValid );
trace( result.ns::Items.ns::TotalResults );
trace( result.ns::Items.ns::Item[0].ns::ASIN );
```


demandware
A Salesforce Company

Once you have created a URL which you can successfully submit using a REST utility, you can then add it to the code which:
Opens the connection
Submits the request
Receives and parses the response.

Here is an example for how to send a request.

Note: There are some great libraries that make it a lot easier, for example Apache's open source HTTPClient.
There are higher level libraries that hide the complexity and manage asynchronous request management for you. These libraries are sometimes created by the platform vendor, and sometimes by third party developers and organizations.

Error Handling

200 (OK)
201 (Created)
204 (No Content)
400 (Bad Request)
401 (Unauthorized)
403 (Forbidden)
404 (Not Found)
405 (Method Not Allowed)
500 (Internal Server Error)

A response will contain a status code. Listed are some of the most common status codes. For a complete list, including details surrounding each one, please see the Demandware documentation.

Sample Faults

HTTP Status Code 400 (Bad Request)

```
{"fault":{"type":"MissingParameterException","message":"parameter 'q' is missing"}}
```

HTTP Status Code 401 (Unauthorized)

```
{"fault":{"type":"LoginException","message":"login failed"}}
```

HTTP Status Code 404 (Not Found)

```
{"fault":{"type":"NotFoundException","message":"resource type required"}}
```

HTTP Status Code 405 (Method Not Allowed)

```
{"fault":{"type":"MethodNotAllowedException","message":"method 'DELETE' not allowed"}}
```

Here are some sample faults.

Sample: Fetch Video Information

```
var videoID : String = args.VideoID,
   youtubeApiUrl : String =
          "https://www.googleapis.com/youtube/v3/videos?part=contentDetails,snippet",
   youtubeApiKey : String = "<YOUR YOUTUBE API KEY>",
   apiCallUrl : String = youtubeApiUrl + "&id=" + videoID + "&key=" + youtubeApiKey;
try {
          var httpClient : HTTPClient = new HTTPClient();
          httpClient.open("GET", apiCallUrl);
          httpClient.send();
          if(httpClient.statusCode != 200) {
                    Logger.error("Invalid http response for the YouTube data request : Error
                    - ({0} - {1})", httpClient.statusCode, httpClient.statusMessage);
          } else {
                    var response : Object = JSON.parse(httpClient.getText()),
                              videoDuration : String =
                                        getDuration(response.items[0].contentDetails.duration);
                    args.VideoData = {};
                    args.VideoData.videoTitle = response.items[0].snippet.title;
                    args.VideoData.videoDuration = (videoDuration) ? (new Date(videoDuration
                              * 1000)).toTimeString().replace(/.*(\d{2}:\d{2}).*/, "$1") : "";
                    args.VideoData.videoImage =
                              response.items[0].snippet.thumbnails['medium'].url;
          }
} catch(e) {
          Logger.error("Error occurred while fetching and parsing YouTube Data. Error -
          ({0})", e);
}
```

This example creates a video page with all the product demo videos from the customer's YouTube account using the YouTube API. It uses the GET method to return video information for given video ID. Notice that the example includes error handling code. Describe the function of this code.

REST-Style Web Services Approach

- Create a secured and public pipeline
- Pass the inventory list ID, product ID and the inventory number to update
- Retrieve inventory list via inventory list ID
- Get the record via product ID
- Update the record
- Respond back via template:

```
<isprint value="UPDATE SUCCESS" encoding="off"/>
```

You can build a custom REST-style web service using a pipeline. This pipeline contains the logic of your REST service and delivers the result via a template. Note: Before creating a custom REST-style web service, check the most recent version of the Open Commerce APIs to see if it contains the functionality that you need. REST services are stateless, and trying to create REST-like service with stateful pipeline calls is not recommended. However, there are times when there is no alternative. When this occurs it is important to limit the number of requests made.

Use case: The client would like to update the inventory in Demandware as soon as the actual inventory changes due to selling something on Amazon or other channel.

Listed are the steps to perform in a typical REST-Style Web Services integration.

Considerations

- Since each request creates a session, limit the number of requests.
- The pipeline only accepts GET/POST requests.
- To make the pipeline callable, set it as public.
- Respond back via a template:

```
<isprint value="${pdict.responseXml}" encoding="off"/>
<isprint value="${pdict.responseJSON}" encoding="off"/>
```

Note the following considerations:
Since each request creates a session, limit the number of requests. The actual number would depend on many factors including: storefront traffic, total number of customer sites, how the realm is provisioned, etc.
The pipeline only accepts GET/POST requests. If you want to implement a full REST pattern, you must provide the REST operation to execute (for example, DELETE resource) as part of the URL.
To make the pipeline callable, set it as public. To do this, have a URL to trigger the pipeline and use the HTTP parameters to pass information in and perform.
Respond back via a template.

Best Practices

- Ensure you have a fully working request and response before coding
- Identify credentialing issues
- Identify data setup issues at the service provider
- There are tool s to help with this
- Paste the working XML requests/responses into the .ds file for reference

Listed are the best practices in a typical REST-Style Web Services integration.

Run a Custom REST-Style Utility

In this lab you will review and run a REST-Style utility.

Use the sample restpipeline.zip, which is available in the Learning Management System as a resource associated with the course.

See additional information in the Course Materials menu.

The utility includes the following pipelines:
Pipeline RESTUtil – Private pipeline that validate HTTP methods and the remote IP address and then extracts JSON from the request object.
Pipeline InventoryAPI-UpdateInventory – Public pipeline (InventoryAPI-UpdateInventory) that takes a JSON request that includes the inventory list ID, product ID, and inventory update count as input. It updates the inventory record and outputs the status as JSON.

The utility also includes the following DW Scripts:
ValidateRemoteIPAddress.ds – Validates remote IP with the allowed IPs configured in _preferences.properties.
CheckRequestMethod.ds – Validates required HTTP method.
ExtractRequestBody.ds – Extracts JSON from the current request object.
UpdateInventory.ds – Get the inventory record for the given product id from the given inventory list and updates with the quantity supplied.

Test the Utility
Review the pipelines in the utility.
Use browser extension like POSTMAN for chrome to test the REST API.
Specify the URL as: https://<your-site-domain>/on/demandware.store/Sites-Avenue-Site/en_US/InventoryAPI-UpdateInventory
Select the HTTP Method as POST.
Supply the request data in following JSON format as raw request data.

```
{
   "inventory_list_id":"mysite-inventory",
   "inventory": {
      "product_id": "04766706",
      "inventoy_update_count": "11"
   }
}
```

Press Send to submit the request.

Additional Considerations

- Demandware suspends pipeline processing after making a request.
- There are a limited number of connections.
- REST service calls are stateless.
- Third-party cartridges available on Xchange.

There are some additional things that you should consider:
Demandware suspends pipeline processing after making a request. Therefore, you will achieve the best performance when the service can process the request in a small amount of time.
Due to a limited number of open connections, avoid long runtimes. It is important to set an appropriate timeout and error handling.
REST service calls are stateless. If you have to integrate with a stateful service, determine where to store the state (for example, in the session or as http parameter, at the basket or customer)
There are many third-party cartridges available on Xchange. You can use these cartridges as a starting point for third party integrations. These cartridges combine the best practice experiences gathered from many projects.

Web Service Framework

- Manages the responsiveness of requests to third party web services
- Ensures use of valid connection timeouts
  - Uses circuit breaker pattern
  - Drop requests when the service is unavailable

To ensure connection timeouts occur, the Web Service Framework uses a circuit breaker pattern.
The circuit breaker pattern ensures that the third party web service isn't inundated with requests after the service has become unresponsive.

Using the Web Service Framework

In this demonstration, you will see:

How to leverage the web service framework to manage the responsiveness of web services.

In addition, review the following for details on web service implementation:

- Demandware documentation
  - https://documentation.demandware.com/DOC2/index.jsp
- Services Manager Best Practices Whitepaper
  - https://xchange.demandware.com/docs/DOC-30539

Play Video

Knowledge Check: 1 of 3

Which of the following are characteristics of a RESTful service integration?

Choose the best answer.

✓	HTTP based architecture
	Uses packets bound via WSDL
	Can be stateful
✓	It has better performance than SOAP
✓	It is language independent

Knowledge Check: 2 of 3

In which format does Demandware send REST responses:

Choose the best answer.

a. JSON.
b. XML.
c. JavaScript.
d. a and b.
e. a, b, and c.

Knowledge Check: 3 of 3

In which of the following languages can you code your REST services:

Choose the best answer.

a. JSON.
b. XML.
c. JavaScript.
d. a and b.
e. a, b, and c.

Next Steps

Now that you understand the basics of REST-style communication, you can apply those same principles to leverage the functionality provided by the Open Commerce APIs.

Now that you understand the basics of REST-style communication, you can apply those same principles to leverage the functionality provided by the Open Commerce APIs.

Open Commerce APIs

The Open Commerce API follows the same methodology as any other REST style communication.

It facilitates one type of real-time server-to-server communication, based on the REST paradigm. You can use any language that can invoke REST-based web services to invoke URLs on the Demandware platform and return information in either XML or JSON. Note: Due to the synchronous nature of real-time services, it works best when the service can process the request in a short timeframe.

It is important to understand what the Open Commerce is not – the developer doesn't use pipelines or script code. Requests are submitted directly to the server. However, the developer can use the Open Commerce API in conjunction with pipeline and script code to create additional site functionality.

04 Open Commerce APIs

Module Objectives

- Describe the purpose and usage of the Open Commerce API.
- Use Business Manager to configure Open Commerce API.
- Apply the RESTful service methodology to the Open Commerce API including: Resources, URLs, and Methods.
- Describe the expected results returned from the Open Commerce API.
- Describe what REST hooks are, how they work, and how to determine where they are located.
- Describe authentication in the Data API.

Upon completion of this module you will be able to:

Describe the purpose and usage of the Open Commerce API.
Use Business Manager to configure Open Commerce API.
Apply the RESTful service methodology to the Open Commerce API including: Resources, URLs, and Methods.
Describe the expected results returned from the Open Commerce API.
Describe what REST hooks are, how they work, and how to determine where they are located.
Describe authentication in the Data API.

demandware
A Salesforce Company

Use Cases: Shop API

- Enable storefront workflows and interactions
- Access storefront resources and data
- Display product ads on external site
- Implement a mobile catalog and shopping application (via DSS)
- Serve content externally (for example, blog)
- Retrieve attributes, HTML, etc.
- Retrieve promotions

The Open Commerce APIs include a Shop API and a Data API. The functionality of these APIs is continuously updated to increase the use cases they support. For the current details, see the documentation.

The Shop API supports the use cases listed.

Use Cases: Data API

- Enable Business Manager-type workflows
- Integrate content management
- Access backend business objects and data (currently limited to customers)
- Customer record search and update (Create, Read, Update, Delete (CRUD))
- Pre-populate account with offsite data
- Integrate with ESP to create customer lists for email and marketing campaigns
- Integrate with the OMS

Note: The Data API is not suitable for mass data.

The Data API supports the use cases listed.

The Bulk API will be an asynchronous data API used to manipulate larger sets of data: Create, Read, Update, Delete (CRUD).

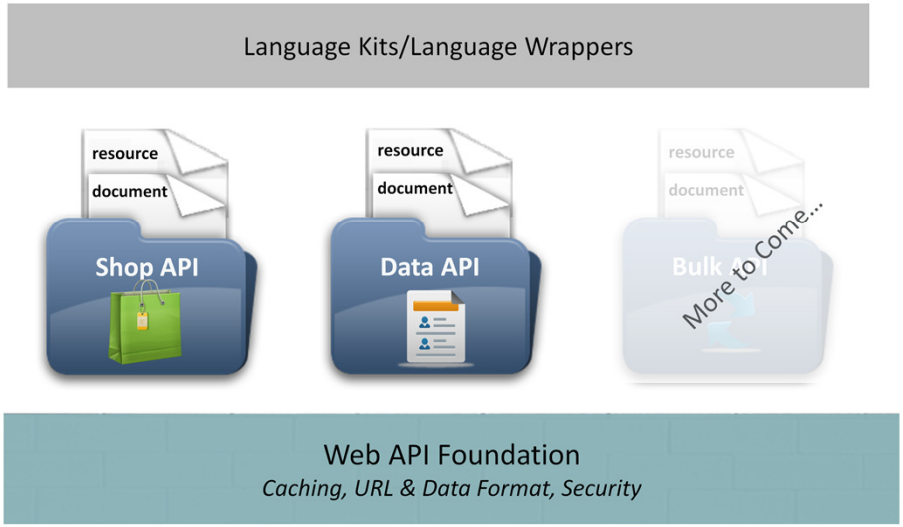
Just Thinking…

Review the documentation to determine the functionality supported by the Shop and Data APIs. Then answer the following questions:

What are some ways that your team can use the Shop API?

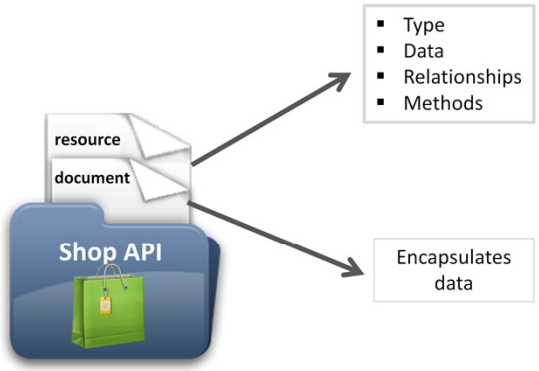
What are some ways that your team can use the Data API?

Open Commerce API Elements



The Open Commerce API gives you the flexibility to use another programming wrapper framework (e.g. PHP, iOS, Java) or mobile language such as Objective C or Java. You can also use JavaScript with AJAX calls and JQuery. Open Commerce API is built upon the web API foundation to provide caching, standard URL & data formats, and security. Each API contains its own document and a resource file. Note: Carefully review documentation for each API to understand the functionality provided as it may increase over time. Additional APIs may be added to the Open Commerce API over time to expand its functionality.

Resources and Documents



As with all RESTful APIs, the resource describes the type, data, relationships, and methods that you can use with the API. The document encapsulates data. Again, you should carefully review the documentation to know what can be exposed through each API.
Documents encapsulate storefront-related data, such as product data or category data. You include them in the body of most API responses. POST, PUT, and PATCH methods typically require documents in the body of the API requests. JSON is the default format for documents, however, you can specify XML as the format.

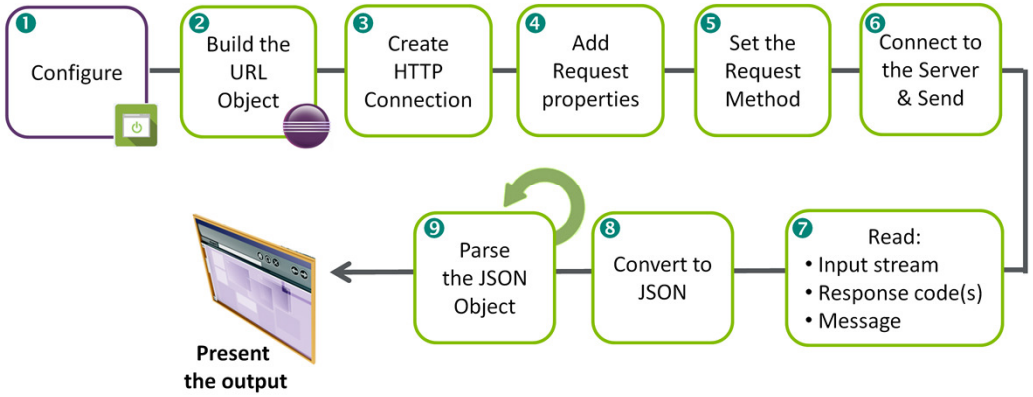
Explore the Open Commerce API Resources and Documents

In this lab, you will explore the elements in the Documents and Resources for each of the APIs in the Open Commerce API.

See the Open Commerce API document in Course Materials for more details.

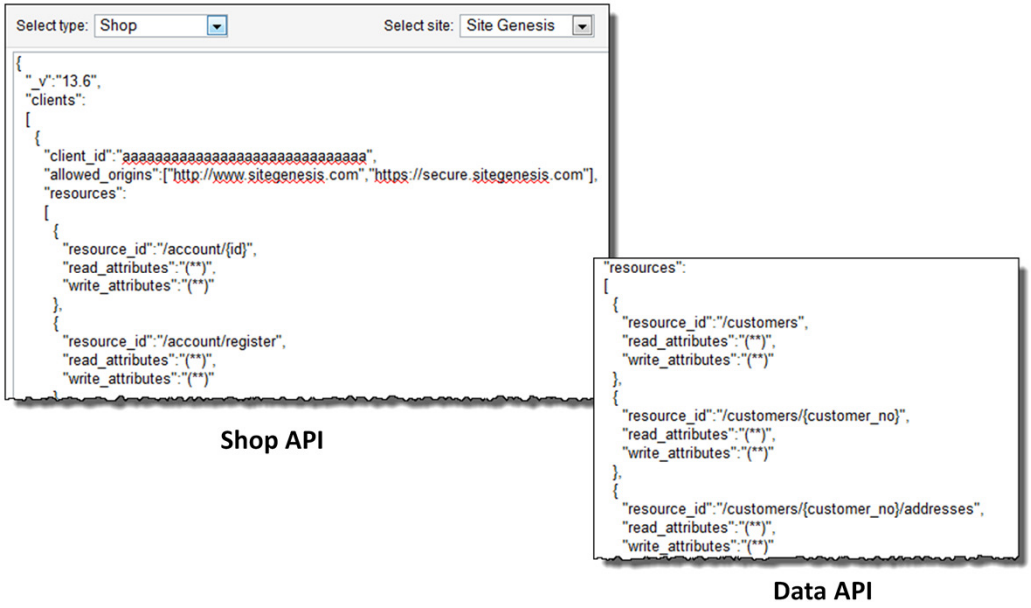
Lab Steps:
Go to the documentation and find the current version of each of the Open Commerce APIs. Answer the following questions.
What is the current version of the Open Commerce API?
What APIs are part of the Open Commerce API?
Which resources do you find for the Shop API? Which do you find for the Data API?
What type of information does each of the resources contain?
What type of information does each of the documents contain?

Process for Invoking Open Commerce API Functionality



The following is a high-level overview of the process for invoking Open Commerce API Functionality:
In the Business Manager, configure the specific API (e.g., Shop API, Data API, etc.) for URL access, version, and permissions.
In eclipse, build the URL object which you will use to invoke REST services.
Create an HTTP connection for the object.
Add request properties to that connection.
Specify the method to submit.
Invoke the HTTP server connection that you previously created to connect to the server and send the request to the server.
Read the input stream from the connection, which is returned in string format. Read the response codes and message from the connection to ensure that the request was successfully processed.
Convert the result to a JSON object for easy parsing.
Parse the JSON object to find items, such as price or image_groups; loop over the items if necessary until all of the desired items are processed.
Present the output to the client.

Configuring the APIs in Business Manager



Shop API

Data API

Before you can use Open Commerce API functionality, you must first configure the Demandware instance of the API using Business Manager. The Business Manager displays a JSON configuration file that you can tailor to your site. If you are using more than one API, such as Shop and Data, then you must configure each one individually.

Go to Administration > Site Development > Open Commerce API Settings to specify the location where to import or create an OCAPI configuration file for the site. Note: Client permissions are cached for up to 3 minute periods before changes become effective.
Select the API type, such as Shop or Data.
Select the site from the list of available sites.
Enter the specifications in the JSON document. Include the following general items: version, client_id, allowed_origins , and response_headers. The JSON document also contains the following resource specifications. Fill them in as appropriate: resource_id, read_attributes, write_attributes, and cache time. For more details, see your documentation.

Example: Configured API

```
{
 "_v":"14.2",
 "clients":
 [
  {
   "allowed_origins":["http://www.sitegenesis.com","https://secure.sitegenesis.com"],
   "client_id":"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa",
   "response_headers":{"x-foo":"bar","P3P":"CP=\"NOI ADM DEV PSAi COM NAV OUR OTR STP IND DEM\""},
   "resources":
   [
    {
     "resource_id":"/product_search",
     "read_attributes":"(**)",
     "write_attributes":"(**)",
     "cache_time":900
    },
    {
     "resource_id":"/products/{id}/set_products",
     "read_attributes":"(c_name,c_street)",
     "write_attributes":"(**)"
    },
    {
     "resource_id":"/basket/{id}/add",
     "read_attributes":"(**)",
     "write_attributes":"(product_id, quantity)"
    }
```

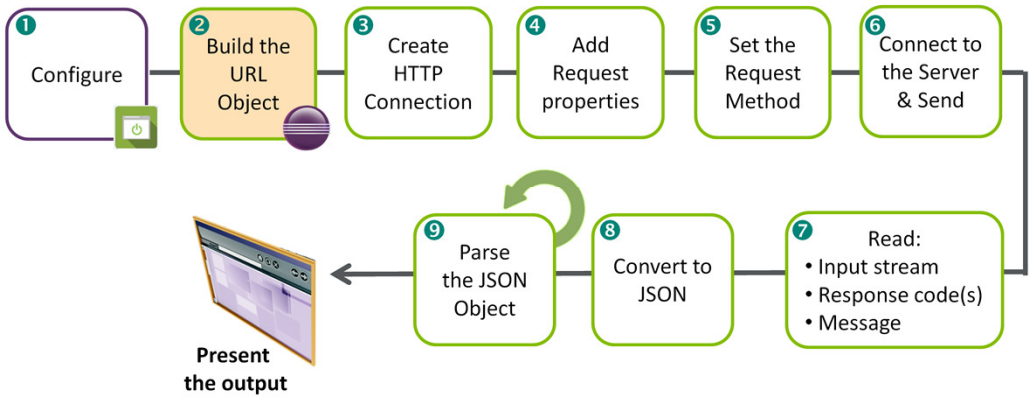
The following is an example configuration.

Configuring How Data Gets Returned

```
{
    "resource_id":"/product_search/images",
    "read_attributes":"(**)",
    "write_attributes":"(**)",
    "config":{
        /* Use the view type "detail" for the property "image" in the document
           ProductSearchHit */
        "search_result.hits.image:view_type":"detail",
        "search_result.variation_attributes.values.image:view_type":"detail",
        "search_result.variation_attributes.values.image_swatch:view_type":"swatch"
    }
```

You can customize how some data is returned. For example, in the Shop API, you can customize how image information is returned by configuring the image view types for certain image properties.

Process for Invoking Open Commerce API Functionality



At this point, we are ready to build the URL object.

Base URL

```
protocol://base_url/version_id/resource_type
```

Production

`https://mysite.com/`

Sandbox

`http(s)://sub_domain.demandware.net/s/site_id/dw/api_type/`

Example:
`http://ajones.inside-na02.dw.demandware.net/s/SiteGenesis/dw/shop`

The base URL provides the main access point to the Open Commence API. It is the same for all API requests. The base URL in production is different from the base URL in a sandbox.

In the production example, mysite.com is the DNS public domain name mapped to the Demandware type. Shop is the API type, but can be data or the name of another Open Commerce API.

In the sandbox example, sub_domain refers to any valid subdomain of the demandware.net domain. For example: http://ajones.inside-na02.dw.demandware.net/s/SiteGenesis/dw/shop

Note: The URL contains an underscore in place of the "." for the version specification.

Extended URL Syntax

Click each item to display the example.

`base_url/version_id/resource_type`

+

`/identifier(s)`
`/action`
`/relationship_type`

The extended URL provides details to support a specific resource and a particular operation. It must conform to the syntax as shown.

Extended URL: Identifier(s)

`base_url/version_id/resource_type`	+	**`/identifier(s)`**
		`/action`
		`/relationship_type`

Single Identifier

```
https://example.com/dw/shop/v14_2/products/008884303989
```

Multiple Identifiers

```
https://example.com/dw/shop/v14_2/products/(333,433,533)
```


demandware
A Salesforce Company

Notice the example URL includes: the protocol, host address, version ID (using an underscore between the year and version number), and resource type (such as products or basket). Additionally you can include a corresponding identifier(s), action, or relationship type as appropriate. They are case sensitive.

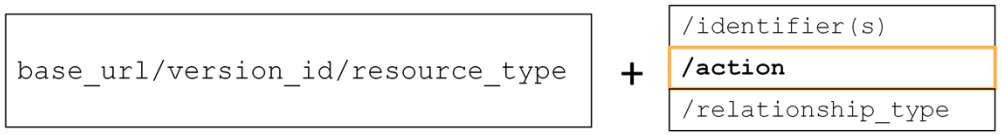
The first example specifies a particular resource instance, in this case, a product SKU.

For some resource types, such as products, you can request multiple resources by specifying multiple identifiers in the request URL, using a comma-delimited list enclosed within parentheses.
The response is returned in the form of an array.

Tip: If the resource type has a plural name, such as products, it provides access to multiple resources.

Note: Use only ASCII characters. You can specify reserved ASCII characters using the % notation. The identifier must be URL encoded. To do this, go to the Business Manager: Site > Site URLs > URL Rules. On the settings tab, in the Blank Substitution field enter the encoding space. For more details, see the documentation.

Extended URL: Action

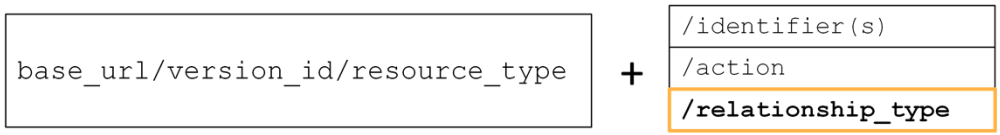


Action

```
https://example.com/dw/shop/v14_2/basket/this/add
```

The action can perform a specified operation, such as login or add a product to the basket. Generally actions are used in conjunction with the HTTP POST method. This example adds a product to a basket. Notice the use of the this identifier, which is a reserved word used to indicate the current resource—in this case, the current basket.

Extended URL: Relationship Type

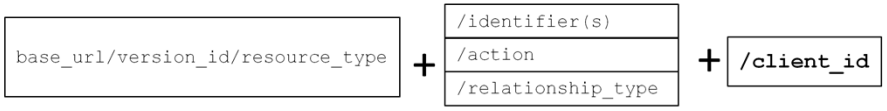


Relationship Type

```
http://example.com/dw/shop/v14_6/
```

The Open Commerce API supports a hierarchy with a maximum of two resources per URL. That is, a resource_type can have a relationship_type in a URL, but a relationship_type cannot also specify yet another relationship_type in a URL.

Extended URL: Client ID

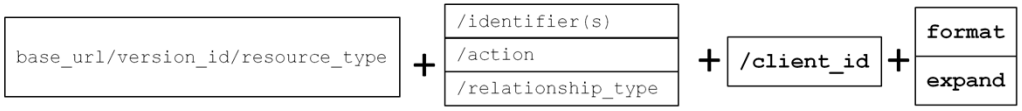


Client ID

```
http://demo.ocapi.demandware.net/s/SiteGenesis/dw/shop/v13_6/
products/008884303989/?client_id=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Next include the client ID, just as you specified in the Business Manager.

Building the Full URL: Other Options



Format

```
https://example.com/dw/shop/v14_2/basket/this/add
?client_id=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa&format=xml
```

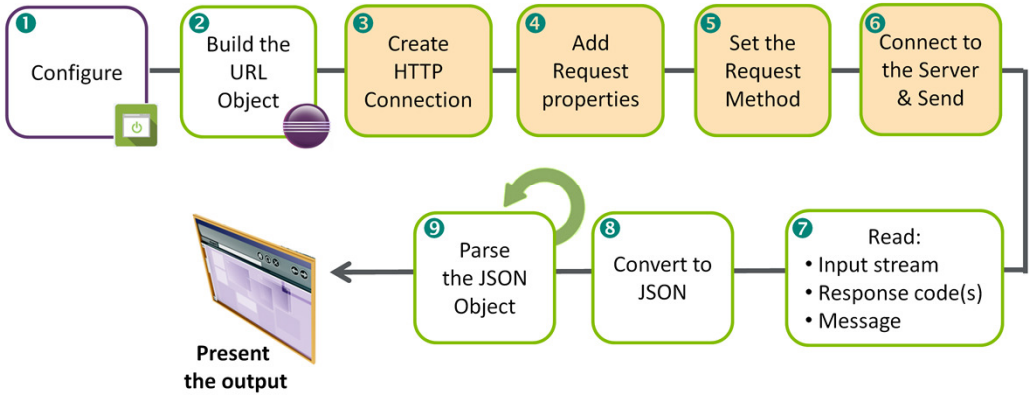
Expand

```
https://example.com/dw/shop/v14_2/products/008884303989?client_id
=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa&expand=images,prices
```

By default, this would return the Site resource document to the browser in JSON format. To return the document in XML format, add the format parameter to the URL. In this example it returns both images and prices in one request.

By default, the results do not return all product data. In the following example there are no links for images to display later. The product resource has an expand parameter, which you can pass in order to return multiple relationship types in one request. In this example it returns both images and prices in one request. This example returns a string containing a comma separated list with the allowed values. Note: It has a maxLength=100.

Process for Invoking Open Commerce API Functionality



At this point in the process, you will perform the following tasks to submit the HTTP request for a particular method:
Create an HTTP connection for the object.
Add request properties to that connection.
Specify the method to submit
Invoke the HTTP server connection that you previously created to connect to the server and send the request to the server.

Just as in any REST style communication, the Open Commerce API supports a standard set of methods, GET, POST, PUT, etc. The methods supported are specific to the API (e.g., Shop API, Data API, etc.).

Creating an HTTP Connection

```
// Open the HTTP Connection
conn = (HttpURLConnection) getURL.openConnection();

//System.out.println(cookies);
conn.setRequestProperty("cookie", cookies);

// Set the HTTP Properties
conn.setRequestMethod("GET");

// Make the HTTP Connection
conn.connect();
```

This example shows how to set up a connection, set the request property, set the request method, and connect to the server.

Using Request Methods

GET

```
REQUEST:
GET /dw/shop/v14_2/products/123 HTTP/1.1
Host: example.com
Accept: application/json

RESPONSE:
HTTP/1.1 200 OK
Content-Length: 67
Content-Type: application/json; charset=UTF-8

{"sku":"123","name":"foo","brand":"bar","online":true}
```

PUT

```
REQUEST:
PUT /dw/merchant/v14_2/products/123 HTTP/1.1
Host: example.com
Content-Type: application/json; charset=UTF-8
Accept: application/json

{"name":"foo","brand":"bar","online":true}

RESPONSE:
HTTP/1.1 201 CREATED
Location: http://example.com/dw/merchant/v14_2/products/123
Content-Length: 0
```


demandware
A Salesforce Company

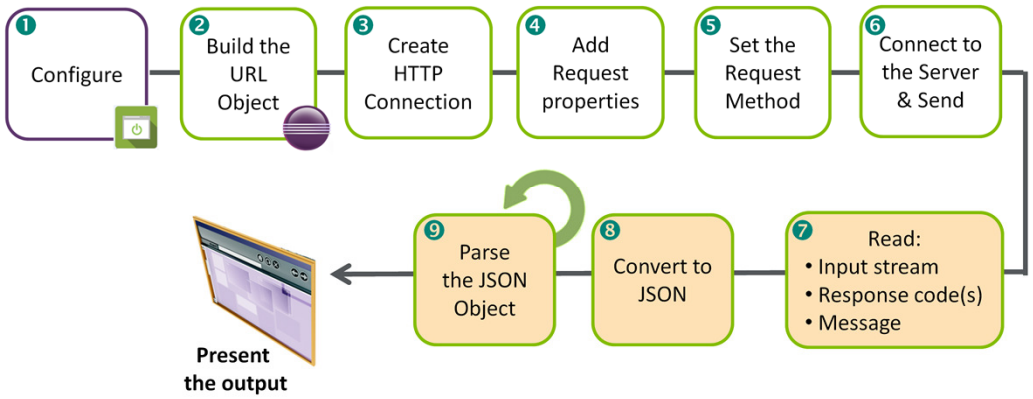
Just as in any REST-style communication, the Open Commerce API supports a standard set of methods, GET, POST, PUT, etc. The methods supported are specific to the API (for example, Shop API, Data API). It is important to review the documentation to determine the supported functionality. What gets specified in the request is dependent on the given method.

The GET example retrieves a Product resource using the Identifier "123". The response has HTTP status code 200, which indicates the resource was found and is contained in the response body.

The PUT example shows how to create a brand new product and contains the "Content-Type" header, which is set to "application/json" plus the charset definition ("UTF-8").

Note: Parameters can include specifications for metadata, localization, cache control, filtering, callback, etc. For a complete list of properties, refer to the documentation.

Process for Invoking Open Commerce API Functionality



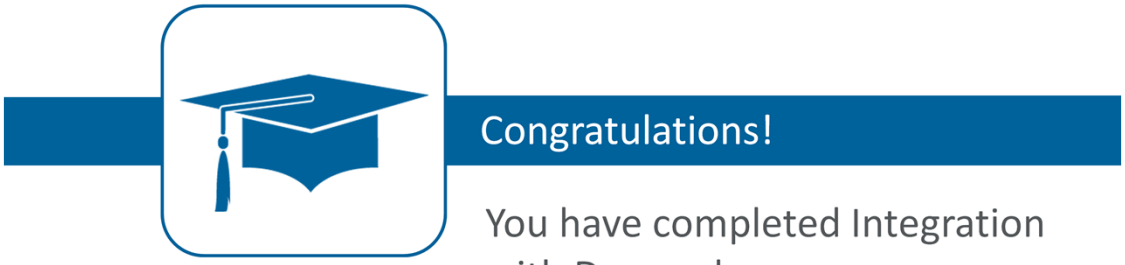
Now that you have built the URL, you can incorporate it into the request and send it to the server for processing.

At this point in the process, you will read the input stream, response codes, and messages.

Open Commerce API

Open the OCAPIexercise document and follow the directions. Sample solutions are available in the Learning Management System as resource associated with the course.

Now that you know the basics for using the Open Commerce APIs, you can leverage them to add functionality to your site.



You have completed Integration with Demandware.