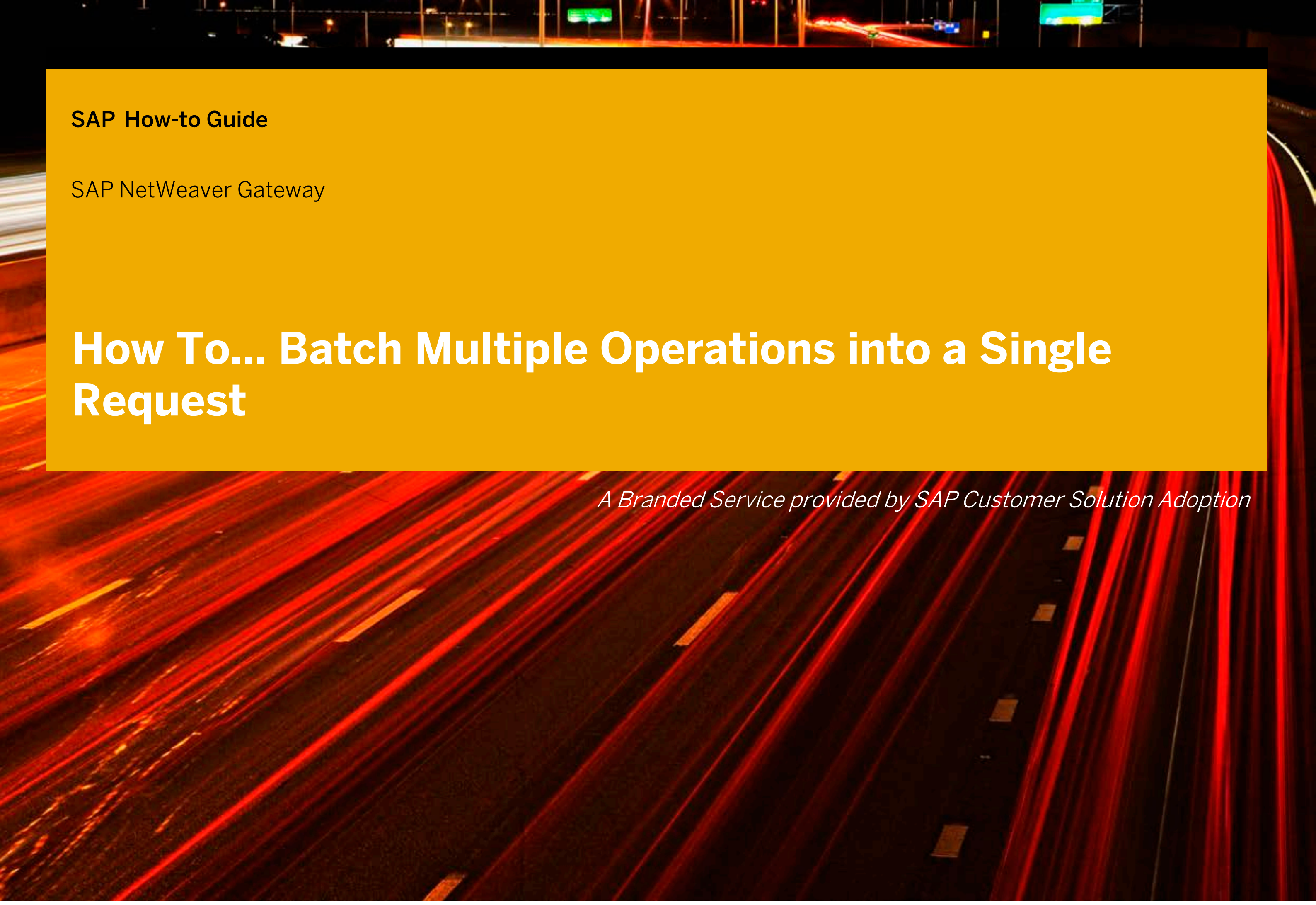


SAP How-to Guide

SAP NetWeaver Gateway

How To… Batch Multiple Operations into a Single Request

A Branded Service provided by SAP Customer Solution Adoption

Applicable Releases:

SAP NetWeaver Gateway 2.0 ≥ SP04

SAP NetWeaver 7.02 ≥ SP07

Version 1.0

June 2012

The Best-Run Businesses Run SAP™

The Best-Run Businesses Run SAP™

Document History

Document Version	Description
1.00	First official release of this guide

Typographic Conventions

Type Style	Description
Example Text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation
Example text	Emphasized words or phrases in body text, graphic titles, and table titles
`Example text`	File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, F2 or ENTER.

Icons

Icon	Description
⚠	Caution
💡	Note or Important
⚙	Example
⬆	Recommendation or Tip

The Best-Run Businesses Run SAP™

Table of Contents

1. Business Scenario

In some cases, business entity instances may logically belong together and need to be handled or processed together in the same logical unit of work. For example, on a sales order, an update of two or more related item entries could be required and must be processed together in a single request (all or none).

SAP NetWeaver Gateway can be used to process such scenarios with its capability to execute multiple operations in a single request.

2. Background Information

If you worked with SAP NetWeaver Gateway in the past, you know that Gateway provides an OData API into an SAP system. This uniform interface supports a given set of operations such as *Create*, *Read*, *Update*, and *Delete* and allows for the processing of a single operation per HTTP request.

However, sometimes it may be required or simply desired for clients to send multiple operations in a single HTTP request. Starting with SP04, SAP NetWeaver Gateway provides the capability for client applications to **batch** multiple operations into a single HTTP request, allowing for a series of retrieve operations and/or modifying operations to be executed using one request.

This document covers how to create and format a batch request and send it to SAP NetWeaver Gateway to be processed.

3. Prerequisites

The following are the prerequisites to complete the steps in this guide:

- You have access to an SAP NetWeaver ABAP 7.02 SP7 or higher system into which the SAP NetWeaver Gateway ABAP add-ons have been installed.
- SAP-delivered RMTSAMPLEFLIGHT service or other Gateway services are available and ready to accept requests.
- Optional (if service data comes from a separate backend): Backend SAP ECC 6.0 or higher with the IW_BEP Gateway add-on installed.
- Optional (if service data comes from a separate backend): Connections between SAP NetWeaver Gateway system and backend SAP application are configured.
- You have at least a basic understanding of ABAP development.
- Google Chrome with "Advanced Rest Client" extension or similar REST client that can send RAW HTTP requests

For more information on related topics, refer to the following:

- SAP NetWeaver Gateway Page on SCN
- SAP NetWeaver Gateway How-To Guides on SCN

More information on OData and Batch processing can be found at http://www.odata.org.

4. Introduction

OData Batch requests allow the grouping of multiple operations into a single HTTP request payload. The components of a batch request, how the request is handled, and the components of the batch response have some significant differences from components and processing of a normal, single-operation OData request.

We start by briefly examining the batch request and response components and how they're handled in Gateway and then show you how to put it together and send a batch request to be processed in Gateway.

4.1 Components of a Batch Request

The main difference between an OData batch request and a normal OData request is that a batch request is represented as a Multipart MIME v1.0 message as specified in RFC2046. This standard format allows multiple parts of various content types to be represented within a single overall request.

4.1.1 Batch Request Header

Batch Requests are submitted as a single HTTP *POST* request to the batch endpoint of a service. The batch endpoint of a service is simply the base service URL appended with "$batch". For example, the batch endpoint for the sample RMTSAMPLEFLIGHT service is:

http://<mygatewayhost>:<port>/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/***$batch***

The batch request must contain a Content-Type header specifying a content type of "multipart/mixed" and a boundary specification. The boundary specification is used in the body of the request to mark the beginning and end of each part of the batch request – i.e. separating the different operations contained in the request.

Due to the use of the HTTP POST method, a CSRF (Cross-Site Request Forgery) token is required when sending the request. For more information on CSRF tokens, please check Cross-Site Request Forgery Protection on the SAP Help Portal.

Below is a sample of what the header of an OData batch request might look like. In this case the boundary marker is specified as "batch", but the use of the term "batch" is not required for the marker. It could just as easily have been "mybatch" or combined with a unique identifier such as a GUID.

```
POST /sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/$batch HTTP/1.1
Host: <mygatewayhost>
Authorization: Basic R1dERU1POmFiY2QxMjM0
X-CSRF-Token: PsvwOGdM3F6hTohiLIFv9A==
Content-Type: multipart/mixed; boundary=batch

<Batch Request Body>
```

4.1.2 Batch Request Body

The body of a batch request is made up of an *ordered* series of retrieve operations and/or change sets. Retrieve operations are basically Query or Read operations executed using the HTTP GET method. Modifying operations are referred to in OData Batch processing terms as "Change Sets".

Change Sets can consist of Create, Update, or Delete operations executed using the HTTP POST, PUT, and DELETE methods. As described in the OData documentation for Batch Processing:

> "A ChangeSet is an atomic unit of work that is made up of an unordered group of one or more of the insert, update or delete operations… ChangeSets cannot contain retrieve requests and cannot be nested (i.e. a ChangeSet cannot contain a ChangeSet)."

Basically, a change set represents a logical unit of work where all changes must be accepted and processed successfully or none of them.

Also from the OData documentation:

> "In the Batch request body, each retrieve request and ChangeSet is represented as a distinct MIME part and is separated by the boundary marker defined in the Content-Type header of the request. The contents of the MIME part which represents a ChangeSet is itself a multipart MIME document with one part for each operation that makes up the ChangeSet."

This is where the boundary defined in the header is used to mark and separate the distinct MIME parts (e.g. to separate each operation).

The OData specification also states that each part representing an operation must include a Content-Type header with value "application/http" and a Content-Transfer-Encoding" header with value "binary".

So let's take a look at a few examples of what the body of a batch request might look like.

Retrieval/Query Example

The following sample batch request shows:

- Two retrieve (Query) operations, one requesting a JSON response.
- Use of boundary marker "batch"

```
--batch
Content-Type: application/http
Content-Transfer-Encoding: binary

GET TravelagencyCollection HTTP/1.1

--batch
Content-Type: application/http
Content-Transfer-Encoding: binary

GET CarrierCollection?$format=json HTTP/1.1

--batch--
```

Notice that only the resource path, relative to the base service URL, is what needs to be provided alongside the method call.

Change Set Example

The following sample batch request shows:

- Two Change Set (Create) operations
- Use of overall batch boundary marker "batch"
- Use of change set boundary marker "changeset"
- Additional Content-Length header specified for Change Set POST parts

```
--batch
Content-Type: multipart/mixed; boundary=changeset

--changeset
Content-Type: application/http
Content-Transfer-Encoding: binary

POST TravelagencyCollection HTTP/1.1
Content-Type: application/atom+xml
Content-Length: ###

<AtomPub representation of Travel Agency 1>***

--changeset
Content-Type: application/http
Content-Transfer-Encoding: binary

POST TravelagencyCollection HTTP/1.1
Content-Type: application/atom+xml
Content-Length: ###

<AtomPub representation of Travel Agency 2>***

--changeset--
--batch--
```

***Note that the actual business data was replaced with a description enclosed in '< >' for clarity.

Retrieval and Change Set Example

The following sample batch request shows:

- Two Queries (beginning and end) and two Change Set operations
- Use of overall batch boundary marker "batch"
- Use of change set boundary marker "changeset"
- Additional Content-Length header specified for Change Set POST parts

```
--batch
Content-Type: application/http
Content-Transfer-Encoding: binary

GET TravelagencyCollection HTTP/1.1

--batch
```

```
Content-Type: multipart/mixed; boundary=changeset

--changeset
Content-Type: application/http
Content-Transfer-Encoding: binary

POST TravelagencyCollection HTTP/1.1
Content-Type: application/atom+xml
Content-Length: ###

<AtomPub representation of Travel Agency 1>***

--changeset
Content-Type: application/http
Content-Transfer-Encoding: binary

POST TravelagencyCollection HTTP/1.1
Content-Type: application/atom+xml
Content-Length: ###

<AtomPub representation of Travel Agency 1>***

--changeset--

--batch
Content-Type: application/http
Content-Transfer-Encoding: binary

GET CarrierCollection?$format=json HTTP/1.1

--batch--
```

***Note that the actual business data was replaced with a description enclosed in '< >' for clarity.

Restriction: Content ID Referencing

As of SAP NetWeaver Gateway 2.0/SP04, the content ID referencing feature as specified in 2.2.1. Referencing Requests in a Change Set is not supported in Gateway.

4.2 Handling of a Batch Request in Gateway

Retrieve operations and change sets are handled differently by Gateway.

Each retrieve operation such as a Query or Read operation within a $batch request will be transferred separately from Gateway to the provider application in the backend system for processing.

On the other hand, since every change set is to be treated as one Logical Unit of Work (LUW), all operations of a change set will be sent at once from Gateway to the provider application in the backend system for processing.

Results from all operations will be collected at the Gateway system and then sent as one HTTP response to the OData consumer.

Application API for $batch

The $batch functionality delivers two additional methods to the /IWBEP/IF_MGW_APPL_SRV_RUNTIME data provider interface.

- CHANGESET_BEGIN
- CHANGESET_END

By default, only one operation per change set is allowed in order to guarantee transactional consistency. If multiple operations are contained in the change set, the default implementation will throw an exception.

In order to process multiple operations in a change set, the default implementation must be overwritten by the application using these methods and the application must further ensure transactional consistency. For example, the application must ensure that a commit or rollback is not triggered while processing the chain of requests contained in the change set.

RMTSAMPLEFLIGHT Service

Note that the RMTSAMPLEFLIGHT sample service is used to test the batch request examples in this guide. If you examine this service, you'll see that the batch methods CHANGESET_BEGIN and CHANGESET_END have been redefined. However, you'll also notice that the redefinitions are virtually empty containing only an "EXIT" statement.

As explained in the comments for the methods, this is sufficient as the modifying methods for all of the entity types does not issue a commit or rollback, so nothing special has to be done to allow for multiple operations in a change set to be processed.

Had there been a commit or rollback that for some reason could not be avoided, the implementation would need to handle this and ensure transactional consistency.

The data provider implementation for RMTSAMPLEFLIGHT can be found within class /IWBEP/CL_MGW_RT_SFLIGHT.

4.3 Components of a Batch Response

Generally, the format of the Batch Response mirrors that of the Batch Request.

Here is an example of a batch response header.

```
HTTP/1.1 202 Accepted
Content-Length: 59882
Server: SAP NetWeaver Application Server / ABAP 702
Content-Type: multipart/mixed; boundary=ejjeeffe0
dataserviceversion: 2.0
```

The HTTP status code "202 Accepted" only indicates that the overall request has been accepted for processing, not that every operation successfully completed. There will be a status code returned for each part of the multipart batch request.

Similar to the batch request, the batch response should also specify a Content-Type of "multipart/mixed" and a boundary specification. As in the case above, the response boundary marker value ("ejjeeffe0") will be different from the value supplied in the request.

The batch response body should contain a response corresponding to each of the operations sent in the batch request. Each corresponding response should be formatted in the same way as if it were processed outside a batch request.

An actual sample response body is shown below:

```
--ejjeeffe0
Content-Type: application/http
Content-Length: 52641
content-transfer-encoding: binary

HTTP/1.1 200 OK
Content-Type: application/atom+xml;type=feed
Content-Length: 52510
dataserviceversion: 2.0
etag: version 01

<AtomPub representation for Travel Agency collection>***

--ejjeeffe0
Content-Type: application/http
Content-Length: 7013
content-transfer-encoding: binary

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 6897
dataserviceversion: 2.0
etag: version 01

<JSON representation for Carrier collection>***

--ejjeeffe0--
```

***Note that the actual business data was replaced with a description enclosed in '< >' for clarity.

5. Step-by-Step Procedure

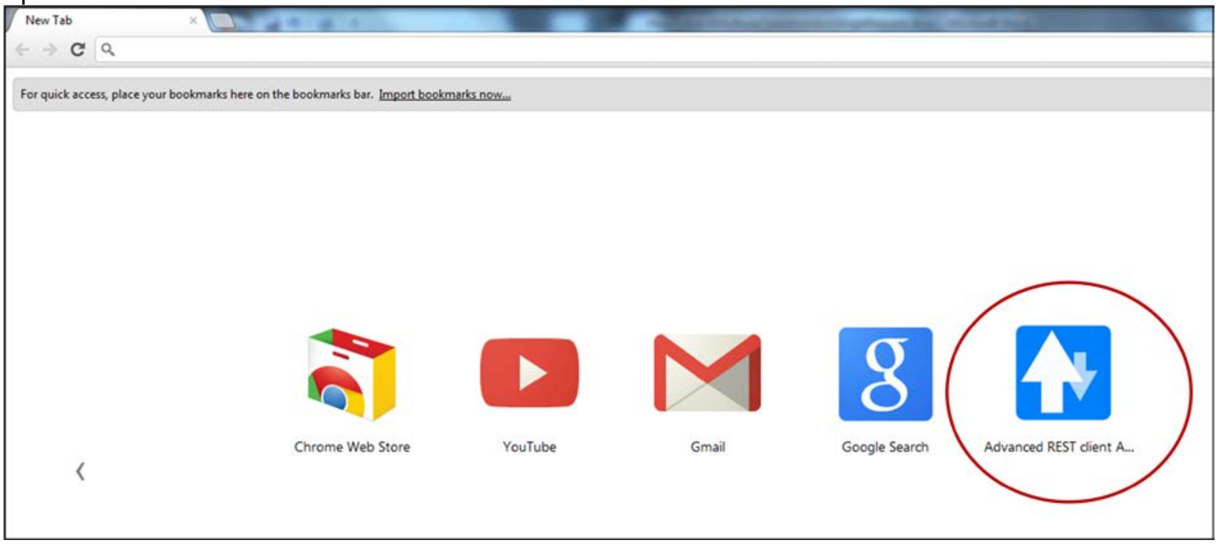
In this guide, the Advanced Rest Client extension for the Google Chrome browser is used to send and test the batch requests. This Advanced Rest Client was chosen due to the relative ease in adding raw input to the body of a batch request. Adding the HTTP body as raw input makes it easier to prepare and send the batch request which can contain multiple operations. However, any REST client that can support the input of a raw HTTP body or support form input of multipart content can be used to send an OData batch request.

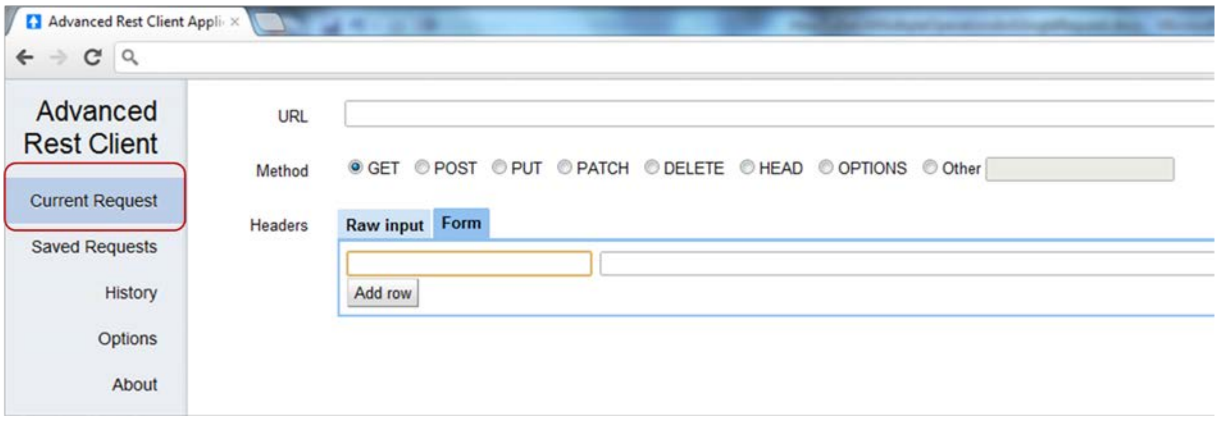
The following steps take you through the steps of setting up and sending a batch request. Once the request header is set up, it's just a matter of differentiating the batch requests in the content of the body.

5.1 Get a CSRF Token

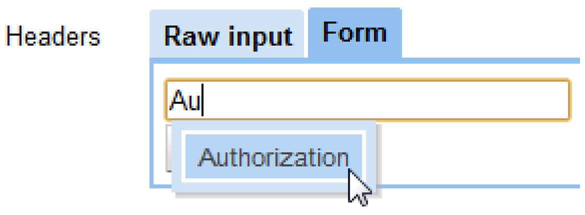
Prior to sending any modifying request such as POST, which $batch uses, we must first retrieve a valid CSRF token so the request is allowed to be processed in Gateway.

1. Open the Advanced Rest Client in Chrome.

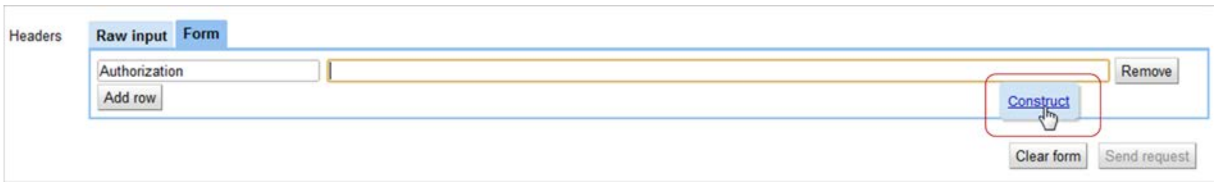

New Tab
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...
Chrome Web Store
YouTube
Gmail
Google Search
Advanced REST client A...



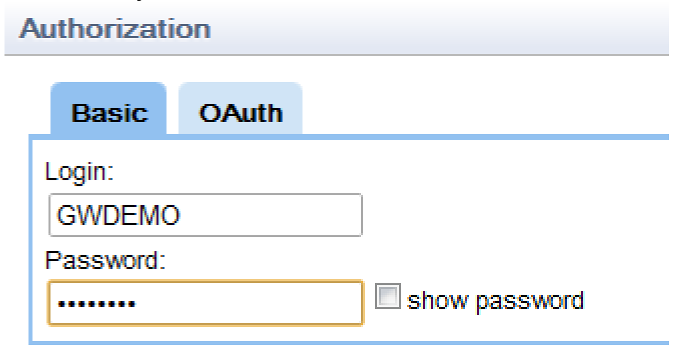
2. Supply user credentials for your Gateway server.
   2.1. In the first, left form field, enter "Authorization" for the HTTP header. As you type, auto completion assistance will be available.



2.2. Now click on the adjacent field on the right and look for a small pop-up showing a link for "Construct" and click on it.



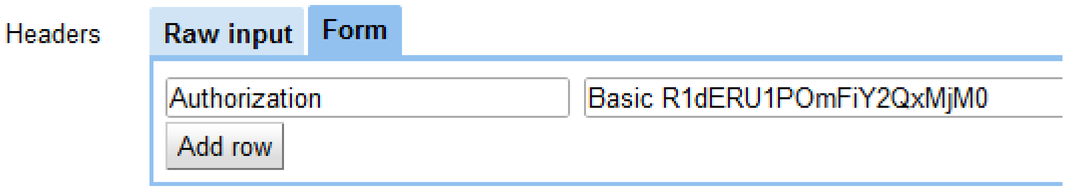
2.3. In the subsequent "Authorization" pop-up, enter your basic authentication user credentials for the Gateway server and click OK.



Login: GWDEMO

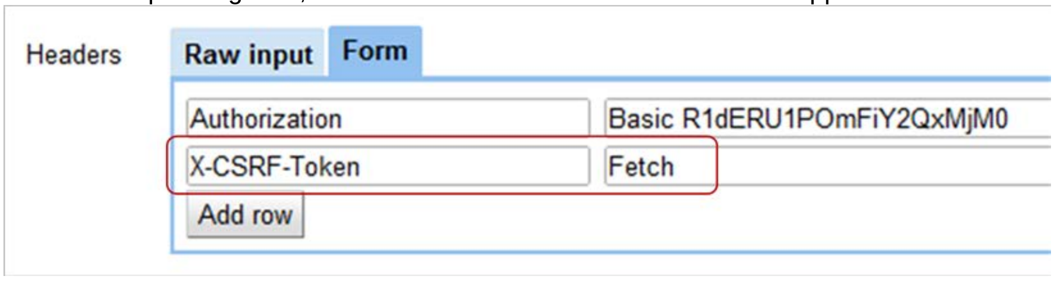
Password: ••••••••

2.4. As a result, your base64-encoded authorization value is displayed for the "Authorization" HTTP header.



Authorization	Basic R1dERU1POmFiY2QxMjM0

3. Add the HTTP header required to retrieve a CSRF (Cross-Site Request Forgery) token.
   3.1. Click on "Add row" to supply the HTTP header for the X-CSRF token.
   3.2. For the header field, enter "X-CSRF-Token". Note that auto completion assistance is unavailable for this header value.
   3.3. In the corresponding field, enter "Fetch". Now the Headers section appears as follows.



Authorization	Basic R1dERU1POmFiY2QxMjM0
X-CSRF-Token	Fetch

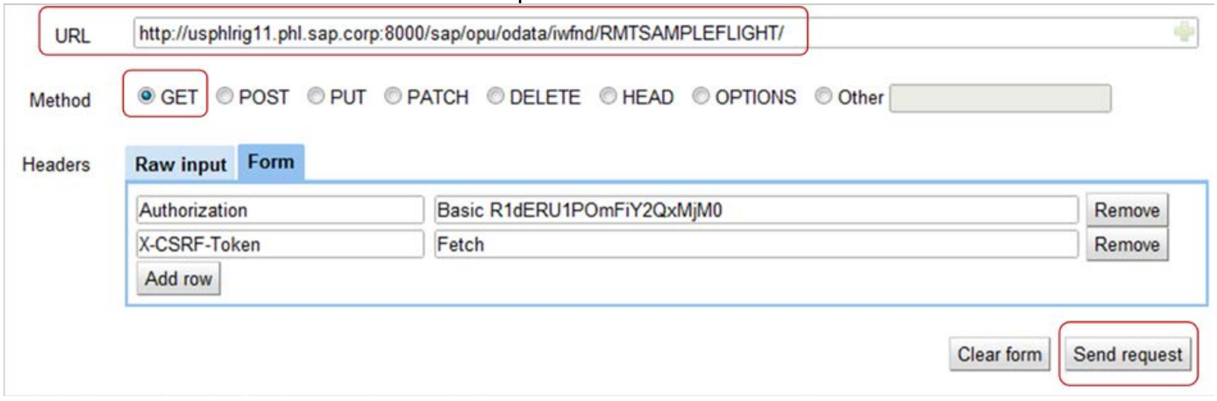
Note

Even if only retrieval operations (e.g. Query or Read) are contained within the batch request, the overall HTTP method of the request is POST. Thus, an X-CSRF token must be obtained and used in the request.

4. For the URL field, enter the base URL for the “RMTSAMPLEFLIGHT” service which is

   http://<mygatewayhost>:<port>**/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/**

5. Select GET for the Method and send the request.

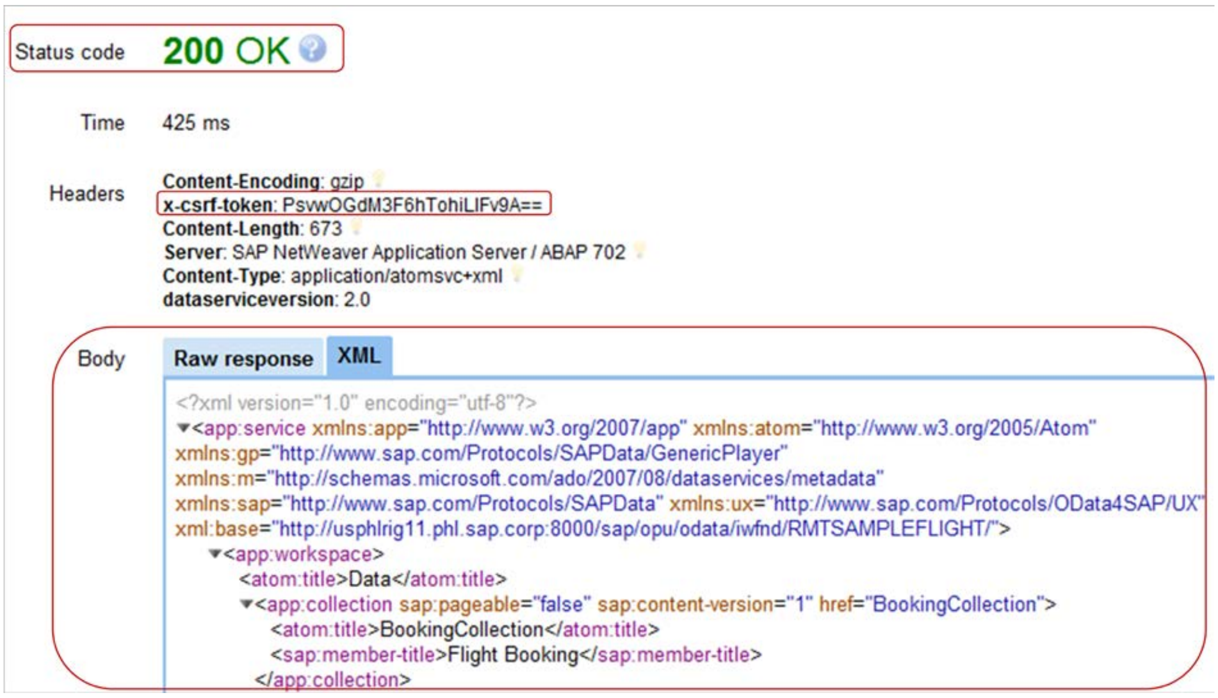


URL	http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/
Method	[✓ GET] POST PUT PATCH DELETE HEAD OPTIONS Other

Headers (Raw input / Form)

Authorization	Basic R1dERU1POmFiY2QxMjM0
X-CSRF-Token	Fetch

6. As a result, you should see a successful response which includes the following:
   - HTTP status code “200 OK”
   - X-CSRF-Token value
   - XML document representing the OData Service Document for the RMTSAMPLEFLIGHT service.



Status code **200 OK**

Time 425 ms

Headers

```
Content-Encoding: gzip
x-csrf-token: PsvwOGdM3F6hTohiLIFv9A==
Content-Length: 673
Server: SAP NetWeaver Application Server / ABAP 702
Content-Type: application/atomsvc+xml
dataserviceversion: 2.0
```

Body (Raw response / XML)

```
<?xml version="1.0" encoding="utf-8"?>
▼<app:service xmlns:app="http://www.w3.org/2007/app" xmlns:atom="http://www.w3.org/2005/Atom"
xmlns:gp="http://www.sap.com/Protocols/SAPData/GenericPlayer"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:sap="http://www.sap.com/Protocols/SAPData" xmlns:ux="http://www.sap.com/Protocols/OData4SAP/UX"
xml:base="http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/">
    ▼<app:workspace>
        <atom:title>Data</atom:title>
      ▼<app:collection sap:pageable="false" sap:content-version="1" href="BookingCollection">
          <atom:title>BookingCollection</atom:title>
          <sap:member-title>Flight Booking</sap:member-title>
        </app:collection>
```

Note

Although in this case the base service URL was used to retrieve the X-CSRF token, the corresponding Query or Read URL (e.g. “...RMTSAMPLEFLIGHT/TravelagencyCollection” or "...RMTSAMPLEFLIGHT /TravelagencyCollection('00000101')" respectively) could also have been used for the same purpose.

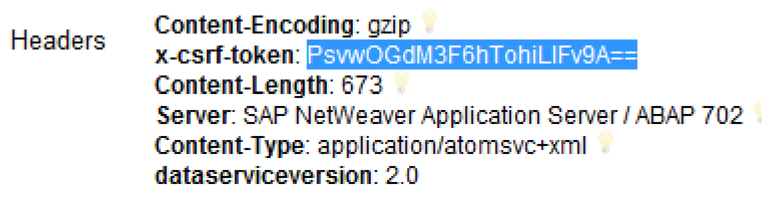
5.2 Complete the Request Header Setup

Now that we have a valid token, supply it in the header and finish setting up the rest of the request header.

1. Modify the URL by appending "$batch" to the end of it.

   http://<mygatewayhost>:<port>/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT***/$batch***

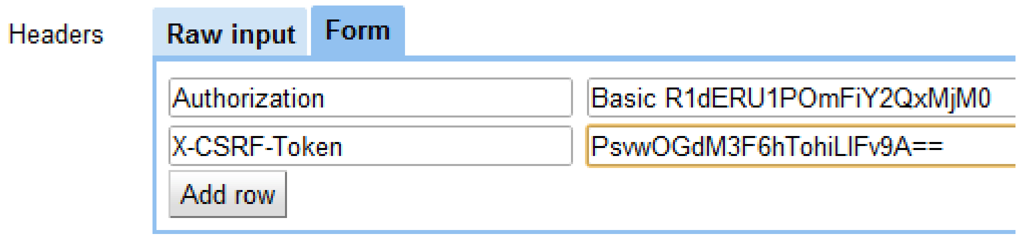
   Batch requests must be submitted as a single HTTP POST request to the "batch endpoint" of a service. The batch endpoint of a service is always the base URL of the service appended by "$batch" as in the sample above.
2. Set the HTTP Method to POST.
3. Set the value for the cross-site request forgery token by copying and pasting the value returned in the response…



Headers

Content-Encoding: gzip
x-csrf-token: PswOGdM3F6hTohiLIFv9A==
Content-Length: 673
Server: SAP NetWeaver Application Server / ABAP 702
Content-Type: application/atomsvc+xml
dataserviceversion: 2.0

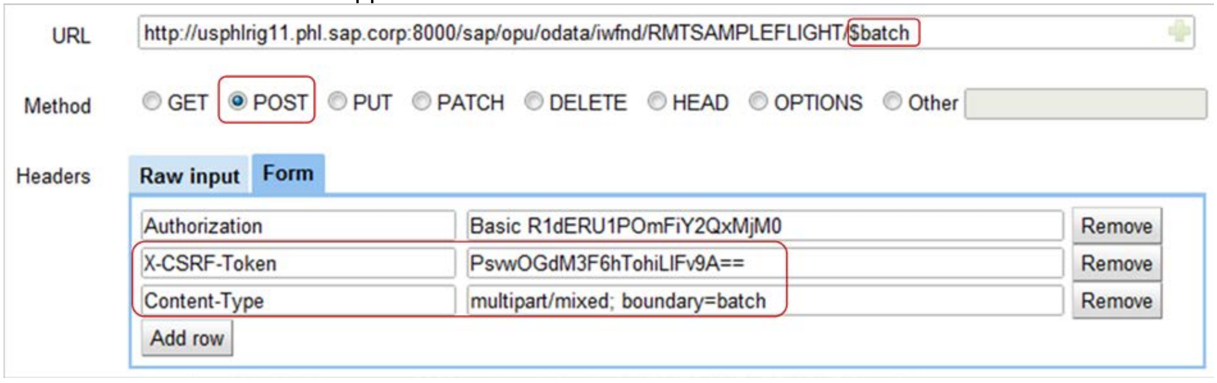
… to the value field for header X-CSRF-Token.



Headers — Raw input | Form

Authorization	Basic R1dERU1POmFiY2QxMjM0
X-CSRF-Token	PswOGdM3F6hTohiLIFv9A==

4. Add HTTP header "Content-Type" with value "mutlipart/mixed; boundary=batch". Now the Advanced Rest Client should appear as follows.



URL: http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/$batch

Method: GET [✓ POST] PUT PATCH DELETE HEAD OPTIONS Other

Headers — Raw input | Form

Authorization	Basic R1dERU1POmFiY2QxMjM0	Remove
X-CSRF-Token	PswOGdM3F6hTohiLIFv9A==	Remove
Content-Type	multipart/mixed; boundary=batch	Remove

5.3 Sending the Batch Request

For a given service, once the headers are set, you should not need to modify them unless the CSRF token expires or you want to use different user credentials.

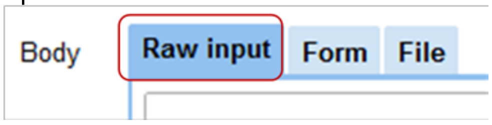
The operations, whether retrieve operations or modifying operations contained in a change set, are defined in the HTTP body.

We will start with by sending a batch request containing only retrieval operations such as Query and Read. Then we'll take a look at change set examples and examples containing a combination of the two.

5.3.1 Retrieval – Query

This section provides an example of executing a batch request containing multiple Read operations.

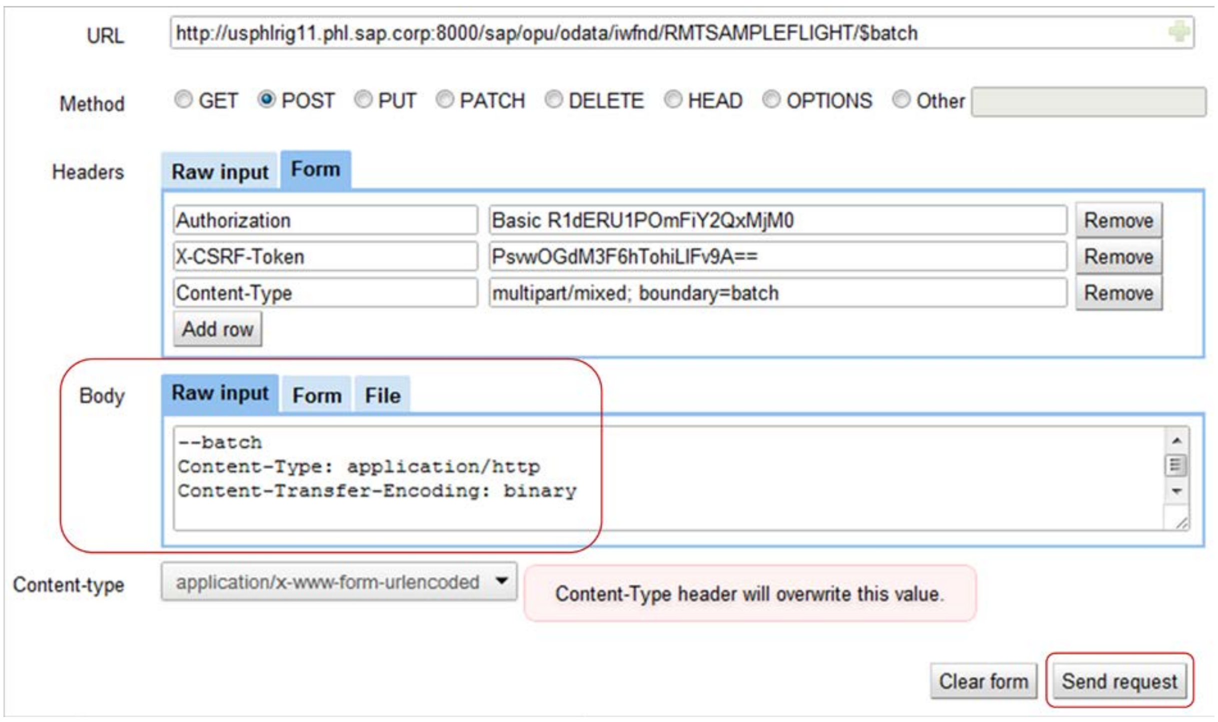
1. In the Advanced Rest client, change how the body of the request will be entered by selecting "Raw input".



2. Supply the following data for the body of the request.
   Due to formatting issues when copying and pasting from a PDF, please use the content from the attached sample file called "Batch_Query.txt".

```
--batch
Content-Type: application/http
Content-Transfer-Encoding: binary

GET TravelagencyCollection HTTP/1.1



--batch
Content-Type: application/http
Content-Transfer-Encoding: binary

GET CarrierCollection?$format=json HTTP/1.1



--batch--
```

3. Send the request.



URL: http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/$batch

Method: GET [✓ POST] PUT PATCH DELETE HEAD OPTIONS Other

Headers (Raw input | Form)

Authorization	Basic R1dERU1POmFiY2QxMjM0
X-CSRF-Token	PswOGdM3F6hTohiLIFv9A==
Content-Type	multipart/mixed; boundary=batch

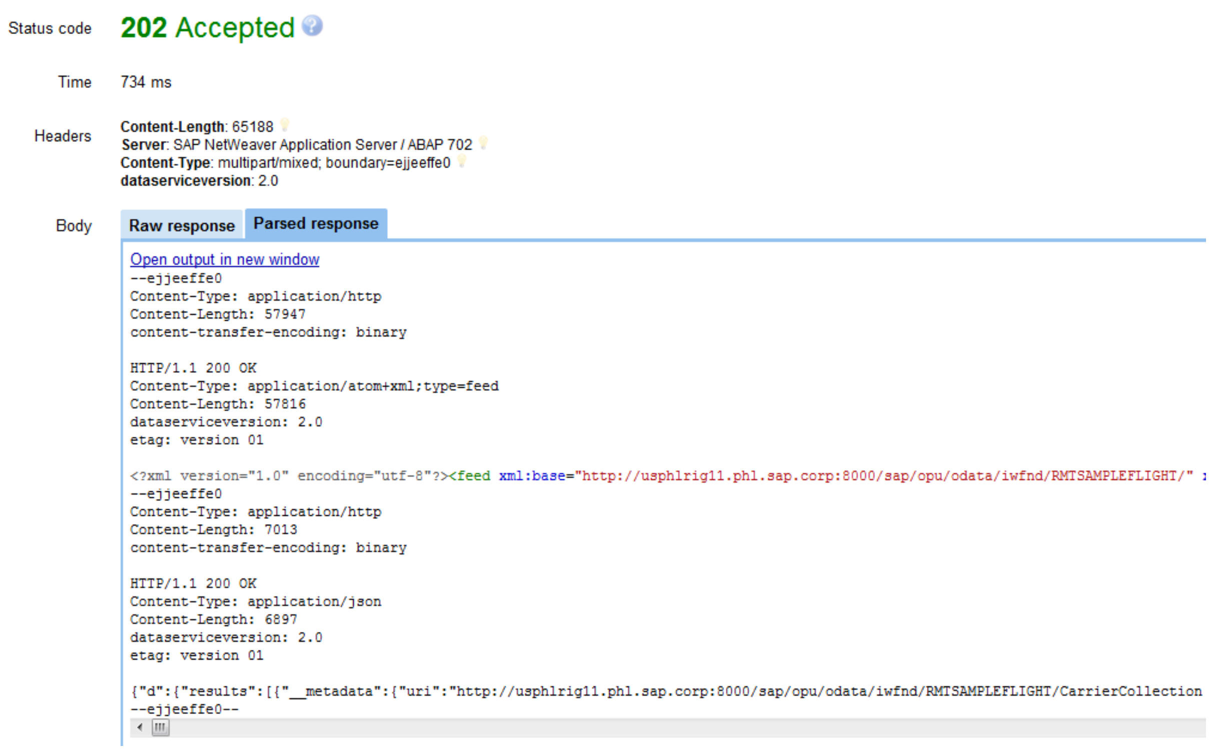
Body (Raw input | Form | File)

```
--batch
Content-Type: application/http
Content-Transfer-Encoding: binary
```

Content-type: application/x-www-form-urlencoded

Content-Type header will overwrite this value.

4. Upon successful execution, you should receive the following:
   - HTTP status code “202 Accepted”.
   - HTTP response code of “200 OK” for each of the operations sent in the request.
   - A response document corresponding to each of the Query operations sent in the request. In this case, one XML response document and one JSON response document.
   - Each response document is formatted in the same way it would have appeared if a single operation were executed outside a batch request.
   - The response documents are returned in the same order as they were sent.



Status code: 202 Accepted

Time: 734 ms

Headers:
Content-Length: 65188
Server: SAP NetWeaver Application Server / ABAP 702
Content-Type: multipart/mixed; boundary=ejjeeffe0
dataserviceversion: 2.0

Body (Raw response | Parsed response)

Open output in new window

```
--ejjeeffe0
Content-Type: application/http
Content-Length: 57947
content-transfer-encoding: binary

HTTP/1.1 200 OK
Content-Type: application/atom+xml;type=feed
Content-Length: 57816
dataserviceversion: 2.0
etag: version 01

<?xml version="1.0" encoding="utf-8"?><feed xml:base="http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/" :
--ejjeeffe0
Content-Type: application/http
Content-Length: 7013
content-transfer-encoding: binary

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 6897
dataserviceversion: 2.0
etag: version 01

{"d":{"results":[{"__metadata":{"uri":"http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/CarrierCollection
--ejjeeffe0--
```

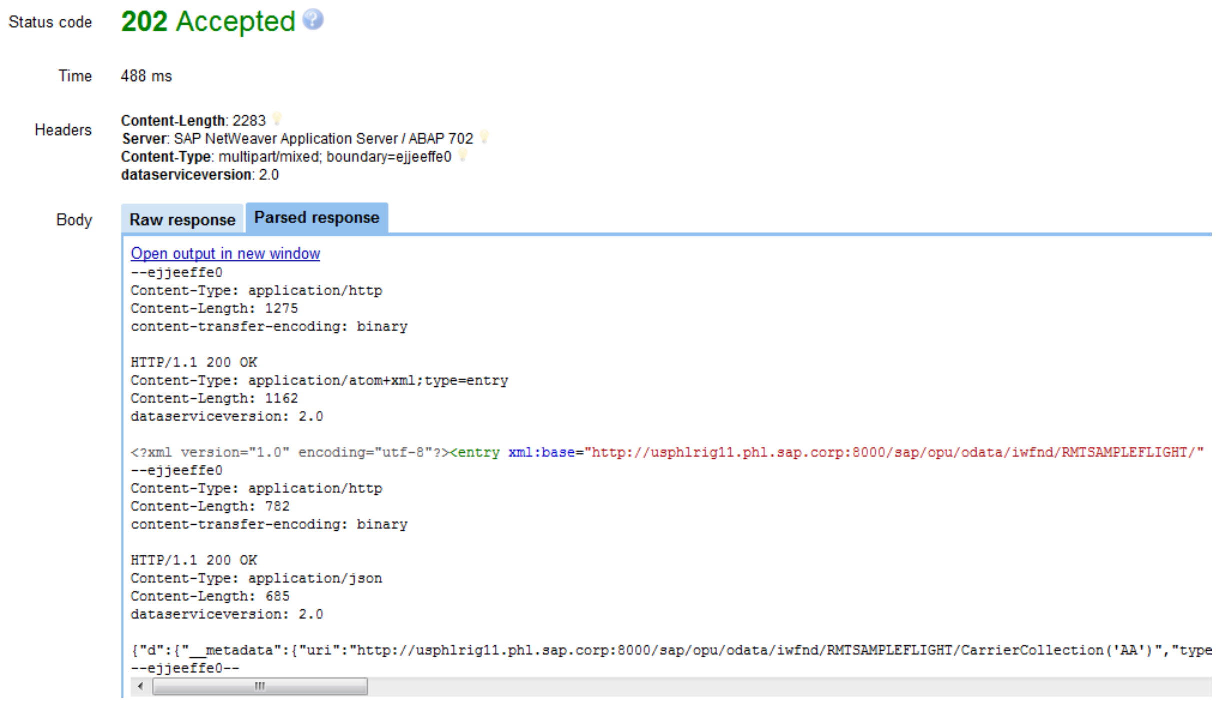
5.3.2 Retrieval – Read

This section provides an example of executing a batch request containing multiple Read operations.

1. Replace the HTTP body of the request with the content below and send the request. Use the content from the attached sample file called “Batch_Read.txt”.

```
--batch
Content-Type: application/http
Content-Transfer-Encoding: binary

GET TravelagencyCollection('00000101') HTTP/1.1



--batch
Content-Type: application/http
Content-Transfer-Encoding: binary

GET CarrierCollection('AA')?$format=json HTTP/1.1



--batch--
```

2. Upon successful execution, you should receive the following:
   - HTTP status code “202 Accepted”.
   - HTTP response code of “200 OK” for each of the operations sent in the request.
   - A response document corresponding to each of the Read operations sent in the request. In this case, one XML response document and one JSON response document.
   - Each response document is formatted in the same way it would have appeared if a single operation were executed outside a batch request.
   - The response documents are returned in the same order as they were sent.



Status code 202 Accepted

Time 488 ms

Headers
Content-Length: 2283
Server: SAP NetWeaver Application Server / ABAP 702
Content-Type: multipart/mixed; boundary=ejjeeffe0
dataserviceversion: 2.0

Body

```
--ejjeeffe0
Content-Type: application/http
Content-Length: 1275
content-transfer-encoding: binary

HTTP/1.1 200 OK
Content-Type: application/atom+xml;type=entry
Content-Length: 1162
dataserviceversion: 2.0

<?xml version="1.0" encoding="utf-8"?><entry xml:base="http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/"
--ejjeeffe0
Content-Type: application/http
Content-Length: 782
content-transfer-encoding: binary

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 685
dataserviceversion: 2.0

{"d":{"__metadata":{"uri":"http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/CarrierCollection('AA')","type
--ejjeeffe0--
```

5.3.3 Change Set – Create

Now we take a look at a change set examples. Remember that operations in a change set should be treated as one logical unit of work where they all succeed or all fail.

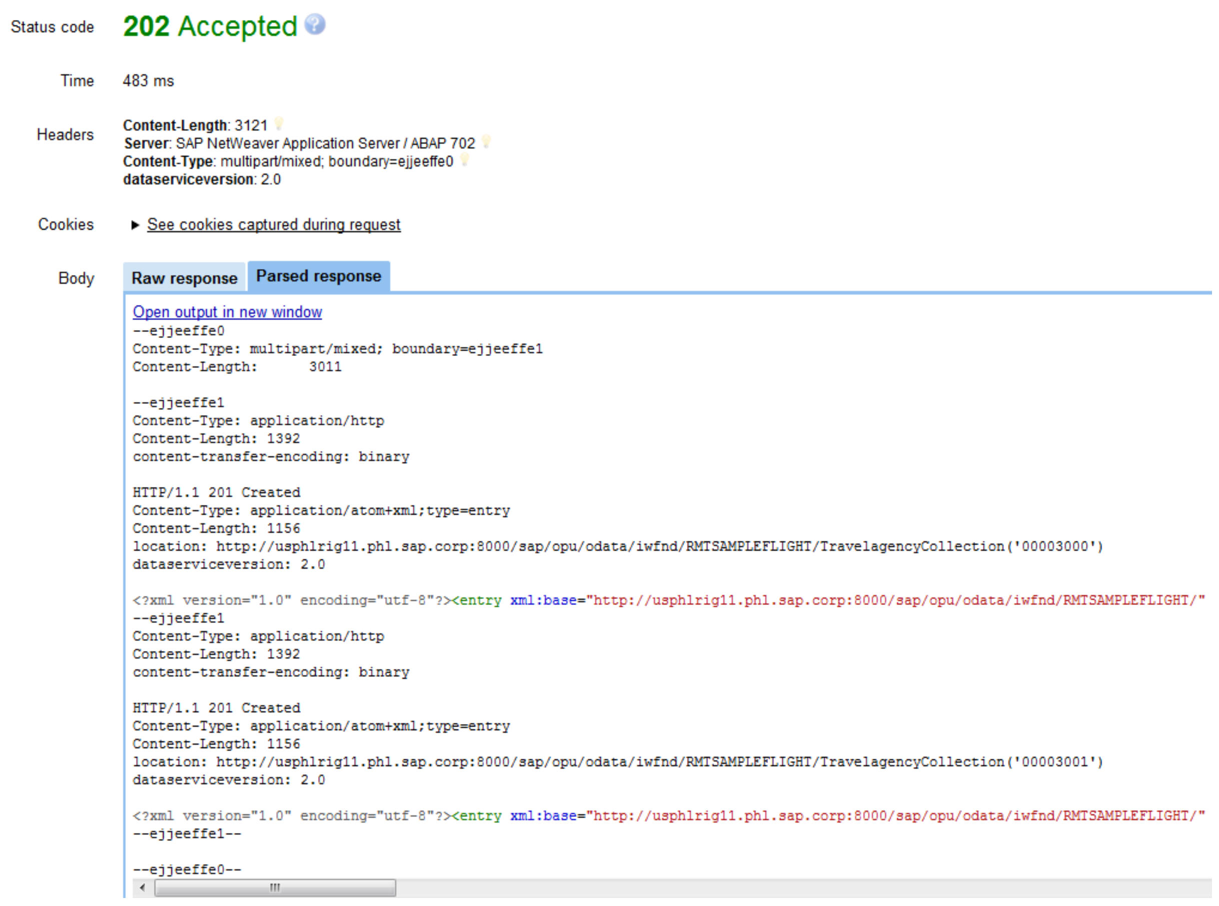
The following is an example of a change set containing multiple Create operations

1. Replace the HTTP body of the request with the content below and send the request.
   Use the content from the attached sample file called "Batch_Create.txt".

```
--batch
Content-Type: multipart/mixed; boundary=changeset

--changeset
Content-Type: application/http
Content-Transfer-Encoding: binary

POST TravelagencyCollection HTTP/1.1
Content-Type: application/atom+xml
Content-Length: 975

<atom:entry xmlns:atom="http://www.w3.org/2005/Atom">
   <atom:content type="application/xml">
      <m:properties

   xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
         xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
         <d:agencynum>00003000</d:agencynum>
         <d:NAME>My Travel Agency</d:NAME>
         <d:STREET>123 Main St.</d:STREET>
         <d:POSTCODE>20004</d:POSTCODE>
         <d:CITY>Washington</d:CITY>
         <d:COUNTRY>US</d:COUNTRY>
         <d:REGION>DC</d:REGION>
         <d:TELEPHONE>+12023123500</d:TELEPHONE>
         <d:URL>http://www.mytravelagency.com</d:URL>
         <d:LANGU>E</d:LANGU>
         <d:CURRENCY>USD</d:CURRENCY>
         <d:mimeType>text/html</d:mimeType>
      </m:properties>
   </atom:content>
</atom:entry>


--changeset
Content-Type: application/http
Content-Transfer-Encoding: binary

POST TravelagencyCollection HTTP/1.1
Content-Type: application/atom+xml
Content-Length: 975

<atom:entry xmlns:atom="http://www.w3.org/2005/Atom">
   <atom:content type="application/xml">
      <m:properties

   xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
         xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
         <d:agencynum>00003001</d:agencynum>
         <d:NAME>My Travel Agency</d:NAME>
         <d:STREET>123 Main St.</d:STREET>
         <d:POSTCODE>20004</d:POSTCODE>
         <d:CITY>Washington</d:CITY>
         <d:COUNTRY>US</d:COUNTRY>
         <d:REGION>DC</d:REGION>
         <d:TELEPHONE>+12023123500</d:TELEPHONE>
         <d:URL>http://www.mytravelagency.com</d:URL>
         <d:LANGU>E</d:LANGU>
         <d:CURRENCY>USD</d:CURRENCY>
```

```
          <d:mimeType>text/html</d:mimeType>
       </m:properties>
    </atom:content>
</atom:entry>



--changeset--
--batch--
```

2. Upon successful execution, you should receive the following:
    - HTTP status code “202 Accepted”.
    - HTTP response code of “201 Created” for each of the operations sent in the request.
    - A response document corresponding to each of the Create operations sent in the request. In this case, since OData services execute a Read upon a successful Create operation, the corresponding Read responses should be displayed.
    - Each response document is formatted in the same way it would have appeared if a single operation were executed outside a batch request.
    - The response documents are returned in the same order as they were sent.


Status code
Time
Headers
Cookies
See cookies captured during request
Body
Raw response
Parsed response
Open output in new window

```
Status code   202 Accepted

Time          483 ms

Headers       Content-Length: 3121
              Server: SAP NetWeaver Application Server / ABAP 702
              Content-Type: multipart/mixed; boundary=ejjeeffe0
              dataserviceversion: 2.0
```

```
--ejjeeffe0
Content-Type: multipart/mixed; boundary=ejjeeffe1
Content-Length:      3011

--ejjeeffe1
Content-Type: application/http
Content-Length: 1392
content-transfer-encoding: binary

HTTP/1.1 201 Created
Content-Type: application/atom+xml;type=entry
Content-Length: 1156
location: http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/TravelagencyCollection('00003000')
dataserviceversion: 2.0

<?xml version="1.0" encoding="utf-8"?><entry xml:base="http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/"
--ejjeeffe1
Content-Type: application/http
Content-Length: 1392
content-transfer-encoding: binary

HTTP/1.1 201 Created
Content-Type: application/atom+xml;type=entry
Content-Length: 1156
location: http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/TravelagencyCollection('00003001')
dataserviceversion: 2.0

<?xml version="1.0" encoding="utf-8"?><entry xml:base="http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/"
--ejjeeffe1--

--ejjeeffe0--
```

Note that you can also verify the changes by proceeding to the backend, executing transaction SE16, and looking at the contents of table STRAVELAG. Entries with key '3000' and '3001' should be created.

3. Now we'll try testing the transactional consistency of the service by trying to create a duplicate Travel Agency record.
   Replace the HTTP body of the request with the content below and send the request.
   Use the content from the attached sample file called "Batch_Create_TransactionalConsistency.txt".

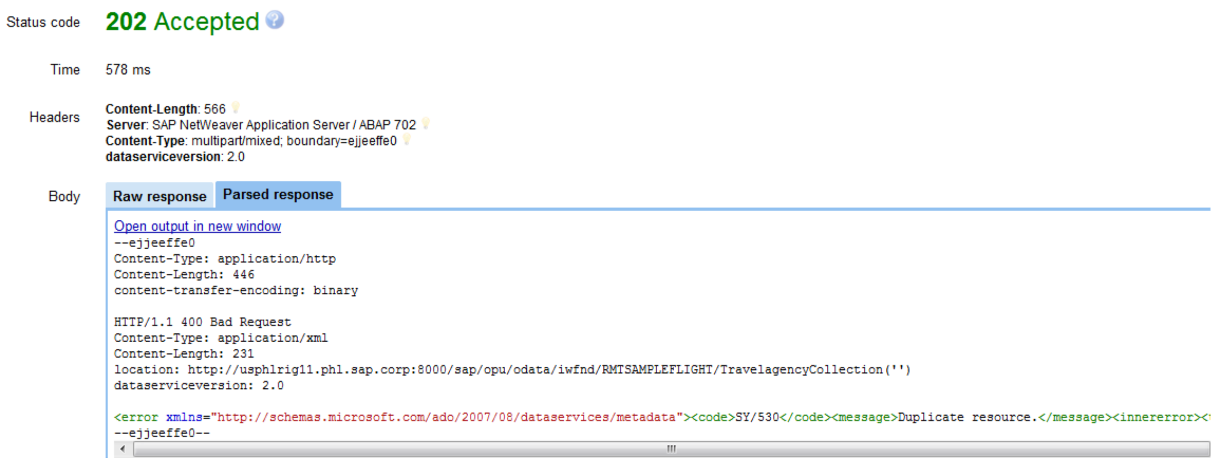
```
--batch
Content-Type: multipart/mixed; boundary=changeset

--changeset
Content-Type: application/http
Content-Transfer-Encoding: binary

POST TravelagencyCollection HTTP/1.1
Content-Type: application/atom+xml
Content-Length: 975

<atom:entry xmlns:atom="http://www.w3.org/2005/Atom">
   <atom:content type="application/xml">
      <m:properties

   xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metada
ta"

   xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
         <d:agencynum>00003002</d:agencynum>
         <d:NAME>My Travel Agency</d:NAME>
         <d:STREET>123 Main St.</d:STREET>
         <d:POSTCODE>20004</d:POSTCODE>
         <d:CITY>Washington</d:CITY>
         <d:COUNTRY>US</d:COUNTRY>
         <d:REGION>DC</d:REGION>
         <d:TELEPHONE>+12023123500</d:TELEPHONE>
         <d:URL>http://www.mytravelagency.com</d:URL>
         <d:LANGU>E</d:LANGU>
         <d:CURRENCY>USD</d:CURRENCY>
         <d:mimeType>text/html</d:mimeType>
      </m:properties>
   </atom:content>
```

```
</atom:entry>



--changeset
Content-Type: application/http
Content-Transfer-Encoding: binary

POST TravelagencyCollection HTTP/1.1
Content-Type: application/atom+xml
Content-Length: 975

<atom:entry xmlns:atom="http://www.w3.org/2005/Atom">
   <atom:content type="application/xml">
      <m:properties

   xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metada
ta"

   xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
         <d:agencynum>00003000</d:agencynum>
         <d:NAME>My Travel Agency</d:NAME>
         <d:STREET>123 Main St.</d:STREET>
         <d:POSTCODE>20004</d:POSTCODE>
         <d:CITY>Washington</d:CITY>
         <d:COUNTRY>US</d:COUNTRY>
         <d:REGION>DC</d:REGION>
         <d:TELEPHONE>+12023123500</d:TELEPHONE>
         <d:URL>http://www.mytravelagency.com</d:URL>
         <d:LANGU>E</d:LANGU>
         <d:CURRENCY>USD</d:CURRENCY>
         <d:mimeType>text/html</d:mimeType>
      </m:properties>
   </atom:content>
</atom:entry>



--changeset--
--batch--
```

Notice that Travel Agency record ‘3002’ would be a new entry, but ‘3000’ already exists as a result of the previous step.

4. Upon execution, you should receive the following:
   - HTTP status code “202 Accepted”.

- HTTP response code of “400 Bad Request” showing an error message indicating “Duplicate resource”.
- No indication that the first/valid Create operation succeeded (you can also verify on the backend that agency ‘3002’ was not created).



Status code **202 Accepted**

Time 578 ms

Headers
Content-Length: 566
Server: SAP NetWeaver Application Server / ABAP 702
Content-Type: multipart/mixed; boundary=ejjeeffe0
dataserviceversion: 2.0

Body

Raw response | Parsed response

Open output in new window

```
--ejjeeffe0
Content-Type: application/http
Content-Length: 446
content-transfer-encoding: binary

HTTP/1.1 400 Bad Request
Content-Type: application/xml
Content-Length: 231
location: http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/TravelagencyCollection('')
dataserviceversion: 2.0

<error xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"><code>SY/530</code><message>Duplicate resource.</message><innererror><
--ejjeeffe0--
```

5.3.4 Change Set – Update

The following is an example of a change set containing multiple Update operations. In this case, this batch request will update two of the records that were created in the previous step.

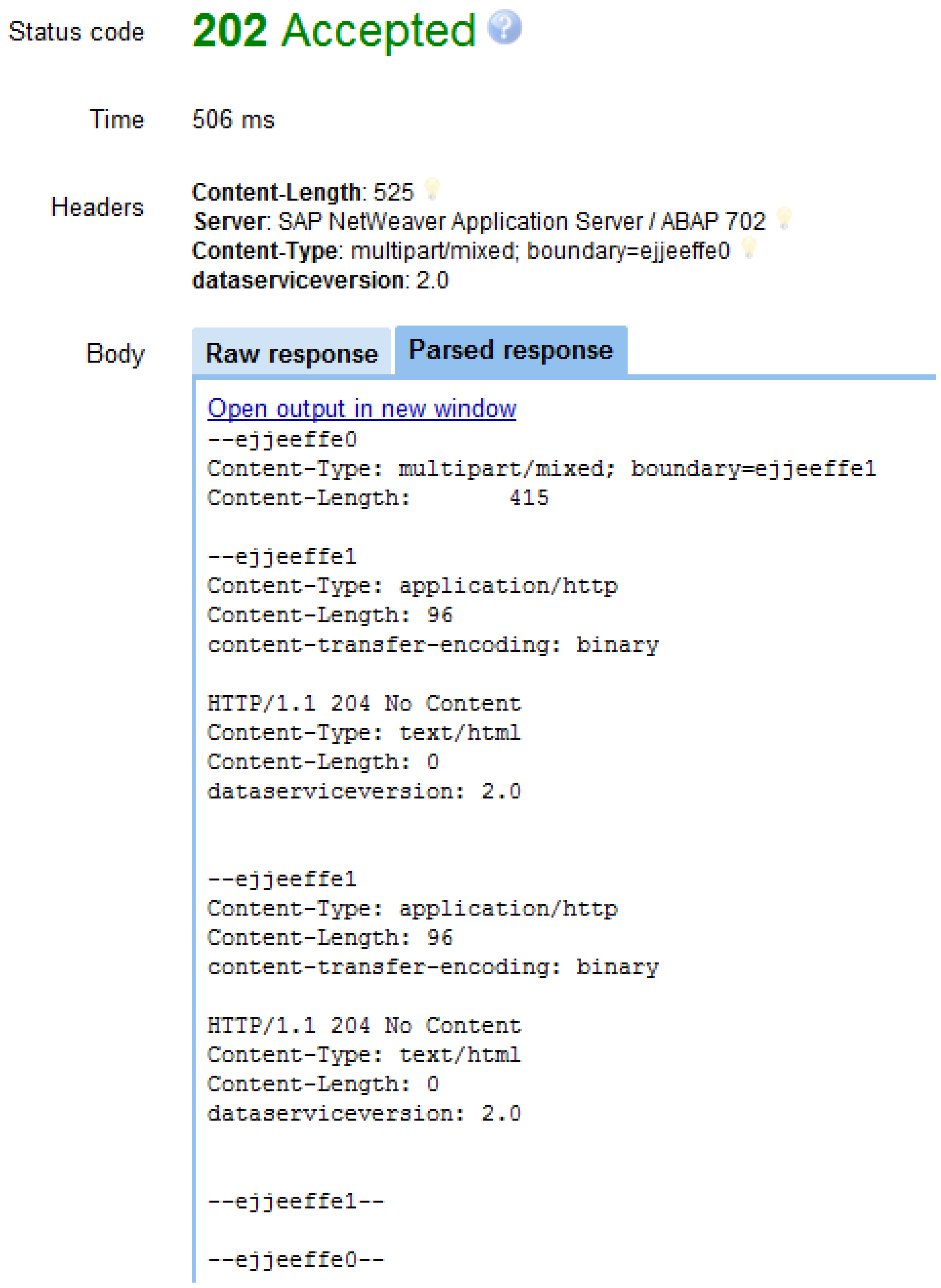
Note that the UPDATE_ENTITY implementation for the RMTSAMPLEFLIGHT service only allows updates for two fields – TELEPHONE and URL. Thus, only these fields will be updated.

1. Replace the HTTP body of the request with the content below and send the request.
   Use the content from the attached sample file called “Batch_Update.txt”.

```
--batch
Content-Type: multipart/mixed; boundary=changeset

--changeset
Content-Type: application/http
Content-Transfer-Encoding: binary

PUT TravelagencyCollection('00003000') HTTP/1.1
Content-Type: application/atom+xml
Content-Length: 975

<atom:entry xmlns:atom="http://www.w3.org/2005/Atom">
   <atom:content type="application/xml">
      <m:properties

   xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
         xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
         <d:agencynum>00003000</d:agencynum>
         <d:NAME>My Travel Agency</d:NAME>
         <d:STREET>123 Main St.</d:STREET>
         <d:POSTCODE>20004</d:POSTCODE>
         <d:CITY>Washington</d:CITY>
         <d:COUNTRY>US</d:COUNTRY>
         <d:REGION>DC</d:REGION>
         <d:TELEPHONE>+1 202 312-3000</d:TELEPHONE>
         <d:URL>http://www.mytravelagency3000.com</d:URL>
         <d:LANGU>E</d:LANGU>
         <d:CURRENCY>USD</d:CURRENCY>
      </m:properties>
   </atom:content>
</atom:entry>

--changeset
Content-Type: application/http
Content-Transfer-Encoding: binary
```

```
PUT TravelagencyCollection('00003001') HTTP/1.1
Content-Type: application/atom+xml
Content-Length: 975

<atom:entry xmlns:atom="http://www.w3.org/2005/Atom">
   <atom:content type="application/xml">
      <m:properties

   xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
         xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
         <d:agencynum>00003001</d:agencynum>
         <d:NAME>My Travel Agency</d:NAME>
         <d:STREET>123 Main St.</d:STREET>
         <d:POSTCODE>20004</d:POSTCODE>
         <d:CITY>Washington</d:CITY>
         <d:COUNTRY>US</d:COUNTRY>
         <d:REGION>DC</d:REGION>
         <d:TELEPHONE>+1 202 312-3001</d:TELEPHONE>
         <d:URL>http://www.mytravelagency3001.com</d:URL>
         <d:LANGU>E</d:LANGU>
         <d:CURRENCY>USD</d:CURRENCY>
      </m:properties>
   </atom:content>
</atom:entry>

--changeset--
--batch--
```

2. Upon successful execution, you should receive the following:
    - HTTP status code “202 Accepted”.
    - HTTP response code of “204 No Content” for each of the operations sent in the request.
    - No corresponding response documents as is expected for Update operations.



Status code **202 Accepted**

Time 506 ms

Headers
Content-Length: 525
Server: SAP NetWeaver Application Server / ABAP 702
Content-Type: multipart/mixed; boundary=ejjeeffe0
dataserviceversion: 2.0

Body

Raw response | Parsed response

Open output in new window

```
--ejjeeffe0
Content-Type: multipart/mixed; boundary=ejjeeffe1
Content-Length:       415

--ejjeeffe1
Content-Type: application/http
Content-Length: 96
content-transfer-encoding: binary

HTTP/1.1 204 No Content
Content-Type: text/html
Content-Length: 0
dataserviceversion: 2.0


--ejjeeffe1
Content-Type: application/http
Content-Length: 96
content-transfer-encoding: binary

HTTP/1.1 204 No Content
Content-Type: text/html
Content-Length: 0
dataserviceversion: 2.0


--ejjeeffe1--

--ejjeeffe0--
```

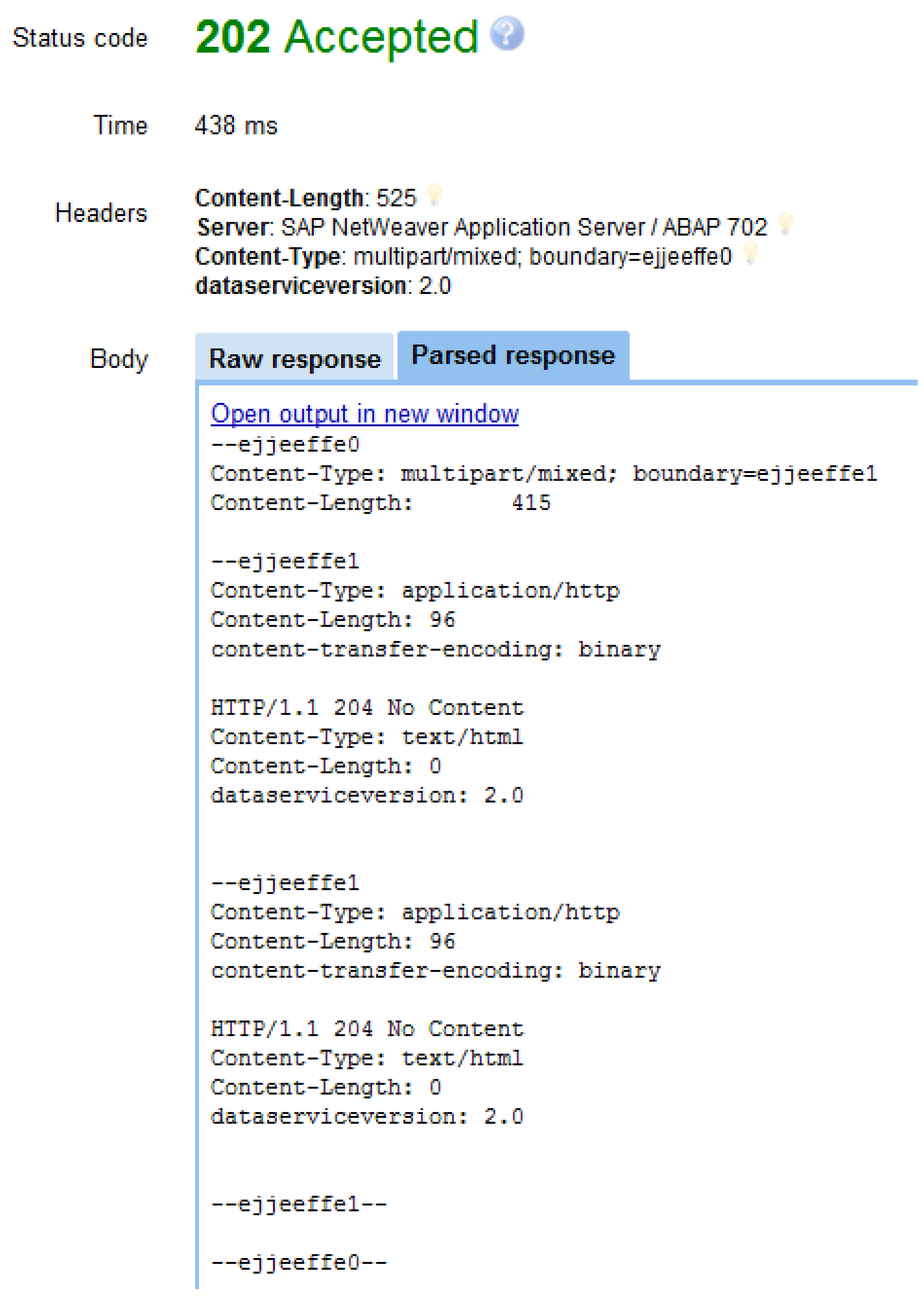
5.3.5 Change Set – Delete

The following is an example of a change set containing multiple Delete operations. In this case, this batch request will delete the two records that were created in a previous step.

1. Replace the HTTP body of the request with the content below and send the request.
   Use the content from the attached sample file called “Batch_Delete.txt”.

```
--batch
Content-Type: multipart/mixed; boundary=changeset

--changeset
Content-Type: application/http
Content-Transfer-Encoding: binary

DELETE TravelagencyCollection(agencynum='00003000') HTTP/1.1



--changeset
Content-Type: application/http
Content-Transfer-Encoding: binary

DELETE TravelagencyCollection(agencynum='00003001') HTTP/1.1



--changeset--
--batch--
```

2. Upon successful execution, you should receive the following:
   - HTTP status code “202 Accepted”.
   - HTTP response code of “204 No Content” for each of the operations sent in the request.
   - No corresponding response documents as is expected for Delete operations.



Status code 202 Accepted

Time 438 ms

Headers
Content-Length: 525
Server: SAP NetWeaver Application Server / ABAP 702
Content-Type: multipart/mixed; boundary=ejjeeffe0
dataserviceversion: 2.0

Body

```
--ejjeeffe0
Content-Type: multipart/mixed; boundary=ejjeeffe1
Content-Length:       415

--ejjeeffe1
Content-Type: application/http
Content-Length: 96
content-transfer-encoding: binary

HTTP/1.1 204 No Content
Content-Type: text/html
Content-Length: 0
dataserviceversion: 2.0


--ejjeeffe1
Content-Type: application/http
Content-Length: 96
content-transfer-encoding: binary

HTTP/1.1 204 No Content
Content-Type: text/html
Content-Length: 0
dataserviceversion: 2.0


--ejjeeffe1--

--ejjeeffe0--
```

5.3.6 Batch Retrieval and Change Set

The following is an example of a change set containing a combination of retrieval operations and a change set. In this case, this batch request will retrieve the collection of Travey Agencies, create two travel agency records in a change set, and finally retrieve the collection of Carriers in JSON format.

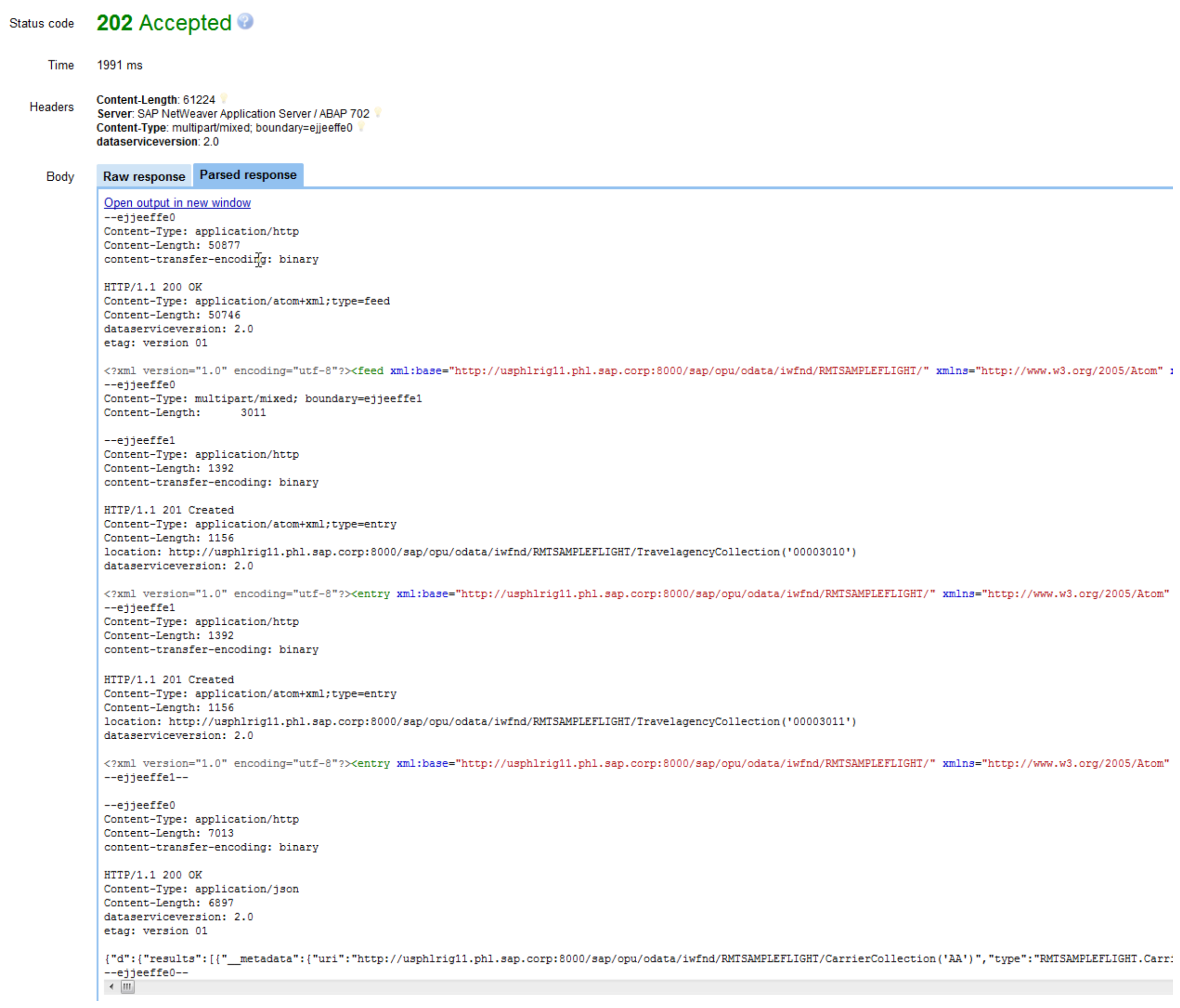
1. Replace the HTTP body of the request with the content below and send the request.
   Use the content from the attached sample file called “Batch_Retrieval_ChangeSet.txt”.

```
--batch
Content-Type: application/http
Content-Transfer-Encoding: binary

GET TravelagencyCollection HTTP/1.1



--batch
Content-Type: multipart/mixed; boundary=changeset

--changeset
Content-Type: application/http
Content-Transfer-Encoding: binary

POST TravelagencyCollection HTTP/1.1
Content-Type: application/atom+xml
Content-Length: 975

<atom:entry xmlns:atom="http://www.w3.org/2005/Atom">
   <atom:content type="application/xml">
      <m:properties

   xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
         xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
         <d:agencynum>00003010</d:agencynum>
         <d:NAME>My Travel Agency</d:NAME>
         <d:STREET>123 Main St.</d:STREET>
         <d:POSTCODE>20004</d:POSTCODE>
         <d:CITY>Washington</d:CITY>
         <d:COUNTRY>US</d:COUNTRY>
         <d:REGION>DC</d:REGION>
         <d:TELEPHONE>+12023123500</d:TELEPHONE>
         <d:URL>http://www.mytravelagency.com</d:URL>
         <d:LANGU>E</d:LANGU>
         <d:CURRENCY>USD</d:CURRENCY>
```

```
          <d:mimeType>text/html</d:mimeType>
       </m:properties>
    </atom:content>
</atom:entry>


--changeset
Content-Type: application/http
Content-Transfer-Encoding: binary

POST TravelagencyCollection HTTP/1.1
Content-Type: application/atom+xml
Content-Length: 975

<atom:entry xmlns:atom="http://www.w3.org/2005/Atom">
    <atom:content type="application/xml">
       <m:properties

    xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
          xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
          <d:agencynum>00003011</d:agencynum>
          <d:NAME>My Travel Agency</d:NAME>
          <d:STREET>123 Main St.</d:STREET>
          <d:POSTCODE>20004</d:POSTCODE>
          <d:CITY>Washington</d:CITY>
          <d:COUNTRY>US</d:COUNTRY>
          <d:REGION>DC</d:REGION>
          <d:TELEPHONE>+12023123500</d:TELEPHONE>
          <d:URL>http://www.mytravelagency.com</d:URL>
          <d:LANGU>E</d:LANGU>
          <d:CURRENCY>USD</d:CURRENCY>
          <d:mimeType>text/html</d:mimeType>
       </m:properties>
    </atom:content>
</atom:entry>


--changeset--


--batch
Content-Type: application/http
```

```
Content-Transfer-Encoding: binary

GET CarrierCollection?$format=json HTTP/1.1



--batch--
```

2. Upon successful execution, you should receive the following:
    - HTTP status code “202 Accepted”.
    - HTTP response code of “200 OK” for each of the Query operations sent in the request (first and last operations).
    - HTTP response code of “201 Created” for each for the operations sent in the request.
    - A response document corresponding to each of the Create operations sent in the request. In this case, since OData services execute a Read upon a successful Create operation, the corresponding Read responses should be displayed.
    - A response document corresponding to each of the Query operations sent in the request. In this case, one XML response document and one JSON response document.
    - Each response document is formatted in the same way it would have appeared if a single operation were executed outside a batch request.
    - The response documents are returned in the same order as they were sent.



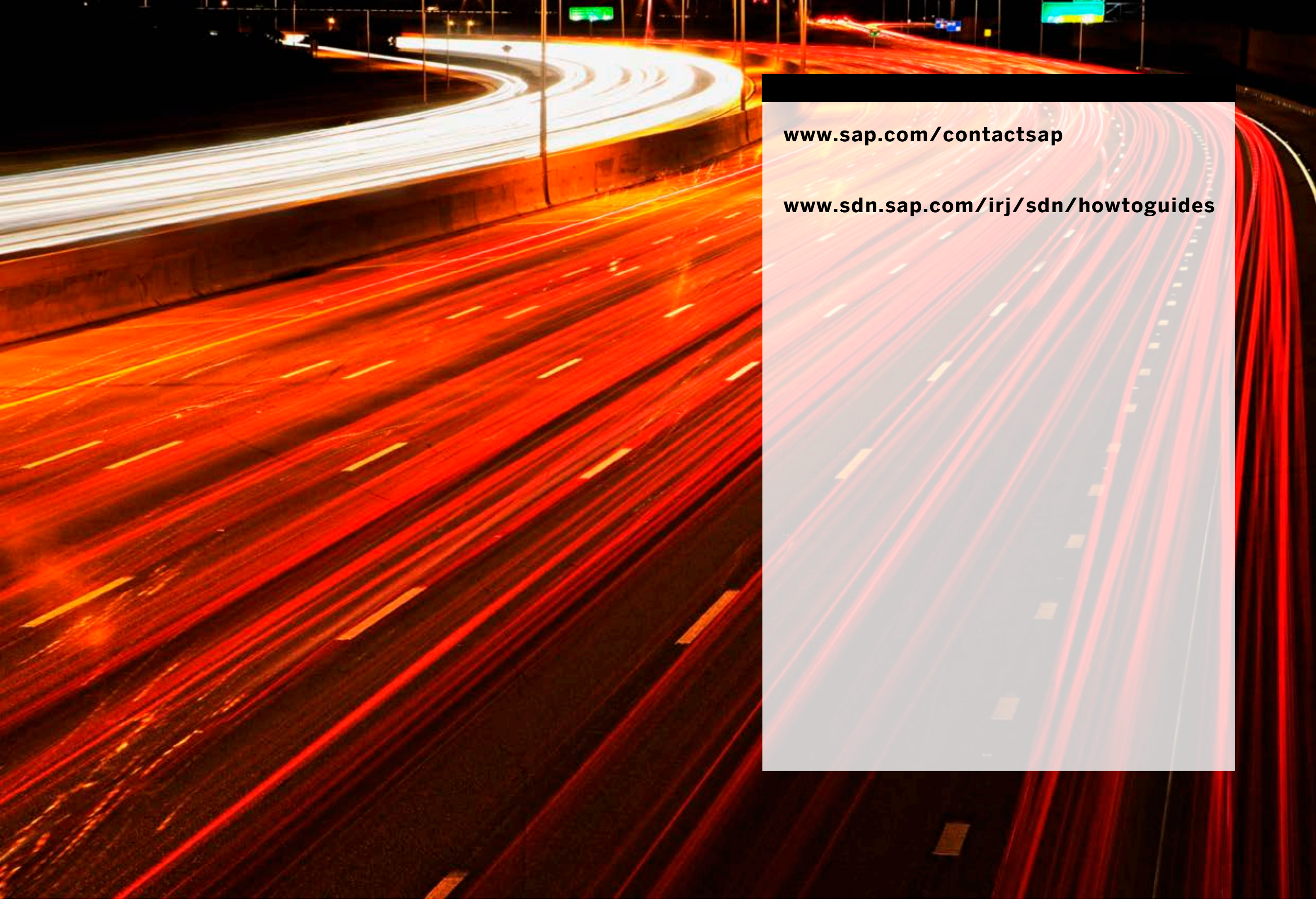
Content-Length: 61224
Server: SAP NetWeaver Application Server / ABAP 702
Content-Type: multipart/mixed; boundary=ejjeeffe0
dataserviceversion: 2.0

```
--ejjeeffe0
Content-Type: application/http
Content-Length: 50877
content-transfer-encoding: binary

HTTP/1.1 200 OK
Content-Type: application/atom+xml;type=feed
Content-Length: 50746
dataserviceversion: 2.0
etag: version 01

<?xml version="1.0" encoding="utf-8"?><feed xml:base="http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/" xmlns="http://www.w3.org/2005/Atom" :
--ejjeeffe0
Content-Type: multipart/mixed; boundary=ejjeeffe1
Content-Length:      3011

--ejjeeffe1
Content-Type: application/http
Content-Length: 1392
content-transfer-encoding: binary

HTTP/1.1 201 Created
Content-Type: application/atom+xml;type=entry
Content-Length: 1156
location: http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/TravelagencyCollection('00003010')
dataserviceversion: 2.0

<?xml version="1.0" encoding="utf-8"?><entry xml:base="http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/" xmlns="http://www.w3.org/2005/Atom"
--ejjeeffe1
Content-Type: application/http
Content-Length: 1392
content-transfer-encoding: binary

HTTP/1.1 201 Created
Content-Type: application/atom+xml;type=entry
Content-Length: 1156
location: http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/TravelagencyCollection('00003011')
dataserviceversion: 2.0

<?xml version="1.0" encoding="utf-8"?><entry xml:base="http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/" xmlns="http://www.w3.org/2005/Atom"
--ejjeeffe1--

--ejjeeffe0
Content-Type: application/http
Content-Length: 7013
content-transfer-encoding: binary

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 6897
dataserviceversion: 2.0
etag: version 01

{"d":{"results":[{"__metadata":{"uri":"http://usphlrig11.phl.sap.corp:8000/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/CarrierCollection('AA')","type":"RMTSAMPLEFLIGHT.Carr:
--ejjeeffe0--
```

6. Summary

Sometimes it may be required or preferred for clients to collect multiple operations in a batch and send them in a single HTTP request. Starting with SP04, SAP NetWeaver Gateway provides the capability for client applications to take advantage of the OData $batch functionality. Using this feature, clients can send multiple operations into a single HTTP request, allowing for a series of retrieve operations and/or modifying operations to be executed at once.

This guide provided an introduction to OData $batch processing, covering the basics of the batch request and response components and how they're handled in Gateway. It also provided details on how to create and format a batch request and send it to SAP NetWeaver Gateway to be processed.



www.sap.com/contactsap

www.sdn.sap.com/irj/sdn/howtoguides

The Best-Run Businesses Run SAP™