NetWeaver Gateway and OData basics
Objectives

Seeing SAP NetWeaver Gateway as a Solution for the Multi-Channel Business
Understanding the OData Protocol
Developing OData Services on SAP ERP
SAP NetWeaver Gateway

A Solution for the Multi-Channel Business
Technology Tipping Point Requires New IT Approach

Support:
End-to-end scenarios
New devices and experiences
Consumer innovations

Growing number of new agile developer communities
Expansion of business data and number of decision makers

Need to provide:
Simple access to complex enterprise systems
Manage and control mission critical systems AND deploy innovative solutions
Enterprise Computing for Business Consumers
Point-to-point Solutions

• Point-to-point solutions often lead to a duplication of both development and administrative effort due to the need to:
  – Develop the same application for multiple mobile operating systems
  – Administer differing app onboarding processes

• The above factors then lead to:
  – Poor scalability
  – Increased system landscape complexity
  – Increased administration effort
Enterprise Computing for Business Consumers
One Data Model → One API → Multiple End-user Experiences

- Any environment, any platform, any experience
- Optimized for user-interaction scenarios
- Suitable for any SAP Business Suite version
- No SAP knowledge required for consumption
SAP NetWeaver Gateway acts as a communication end point.

**IMPORTANT**
Gateway service development takes place *in whichever system contains the IW_BEP add-on.*

However, **IW_BEP** can also be installed in the SAP Business Suite system; in which case, Gateway service development takes place directly in the backend Business Suite System.
SAP NetWeaver Gateway

Origins of the OData Protocol
What is the Open Data Protocol (OData)?

OData is an open standard originally developed by Microsoft, but now managed by the [Oasis Organisation](http://www.oasis-open.org). It is based on the Atom Publishing and Atom Syndication standards, which in turn, are based on XML and HTTP(S).

It was designed to provide a standardised implementation of a RESTful API. In doing so, it offers database-like access to server-side resources. Hence, OData has been described as: “ODBC for the Web”

OData is also extensible. This allows SAP to supplement the data types used by OData with extra information from the ABAP Data Dictionary.
What Does OData Add to Atom?

The Atom Publishing Format does not specify how data should be encoded within a Feed; therefore, it fails to be fully RESTful because its messages are not self-describing.

OData extends Atom by providing a metadata description of the message.

OData provides definitions for:
- Simple Types
- Complex Types
- Associations between entries
- Navigation Paths between entries
- Custom behaviour (known as function imports) beyond the standard QUERY, CREATE, READ, UPDATE, DELETE (QCRUD) operations
Output Formats Supported by OData

XML

```xml
<feed xmlns="http://www.w3.org/2005/Atom"
                 /GBAPP_POAPPROVAL.WorkflowTaskCollection">
                  WorkflowTaskCollection</id>
  <title type="text">WorkflowTaskCollection</title>
  <updated>2013-06-19T07:03:52Z</updated>
  <author>
    <name/>
  </author>
  <link href="WorkflowTaskCollection" rel="self">
                  WorkflowTaskCollection</id>
    <title type="text">WorkflowTaskCollection</title>
    <title type="text">WorkflowTaskCollection(SAP__Origin = "IDM_800_RIG")</title>
  </link>
  <entry>
                  WorkflowTaskCollection(WorkflowTaskID = "I003866")</id>
    <title type="text">WorkflowTaskCollection(SAP__Origin = "IDM_800_RIG")</title>
    <content type="application/xml">
      <m:properties>
        <d:SAP__Origin>IDM_800_RIG</d:SAP__Origin>
        <d:TaskType>TS20000166</d:TaskType>
        <d:PoNumber>4500017386</d:PoNumber>
        <d:WiCreatedAt>2013-06-05T08:53:10</d:WiCreatedAt>
      </m:properties>
    </content>
  </entry>
</feed>
```

JSON

```json
{
  "d": {
    "results": [
      {
        "__metadata": {
          "type": "GBAPP_POAPPROVAL.WorkflowTask"
        },
        "SAP__Origin": "IDM_800_RIG",
        "WorkitemID": "000001105838",
        "TaskType": "TS20000166",
        "PoNumber": "4500017386",
        "PoNumberFormatted": "4500017386",
        "WiCreatedAt": "/Date(1370422390000)/",
        "ForwardedByID": "",
        "ForwardedByName": "",
        "SubstitutingForID": "",
        "SubstitutingForName": "",
        "CreatedByID": "I003866",
        "CreatedByName": "I003866",
        "Value": "1023.00",
        "Currency": "EUR",
        "SupplierID": "1000",
        "SupplierName": "C.E.B. BERLIN"
      }
    ]
  }
}
```
SAP NetWeaver Gateway

Exposing Entity Data Models Using the OData Protocol
An **Entity Data Model** (EDM) is the starting point when designing an OData service. The EDM describes the organisation and relationship of the resources within a particular business scenario.

- **Entity Type**: EDM basic building block. Represents specific business object.
- An Entity Type is built from one or more properties.
- **Navigation**: The runtime implementation of an association.
- **Function Import**: Additional operations not covered by CRUD.
- **At least one property must be nominated as a key field**.

**Association**: Declares that a relationship exists between two entity types.
OData Service Documents – The Consumption Starting Point

After an Entity Data Model has been turned into an OData Service, the Gateway system will provide you with a URL to access this service.

When this URL is requested, the Gateway service will respond by sending you the Service Document.

The Service Document is a high-level description of the resources exposed by the OData service.

```xml
<app:service xmlns:app="http://www.w3.org/2007/app"
             xmlns:atom="http://www.w3.org/2005/Atom"
             xmlns:sap="http://www.sap.com/Protocols/SAPData" xml:lang="en"
  <app:workspace>
    <atom:title type="text">Data</atom:title>
    <app:collection sap:creatable="false" sap:updatable="false"
                    sap:deletable="false" sap:addressable="false" sap:content-version="1"
                    href="ItemDetailCollection">
      <atom:title type="text">ItemDetailCollection</atom:title>
      <sap:member-title>ItemDetail</sap:member-title>
    </app:collection>
  </app:workspace>
  <app:collection sap:requires-filter="true" sap:content-version="1"
                  href="UserDetailsCollection">
    <atom:title type="text">UserDetailsCollection</atom:title>
  </app:collection>
</app:service>
```
Obtaining OData Metadata

In order to consume an OData service, you should retrieve the metadata that describes the service. This is done by adding the suffix `$metadata` to the Service Document URL. (OData metadata is only available in XML format, not JSON)


This will now return an **Entity Data Model** (edmx) XML description of the service:

```xml
   <edmx:DataServices m:DataServiceVersion="2.0">
      <Schema xmlns="http://schemas.microsoft.comado/2008/09/edm" Namespace="gbapp_p">
         + <EntityType Name="Subscription" sap:semantics="subscriptions" sap:content-version="1">
         + <EntityType Name="Notification" sap:semantics="notifications" sap:content-version="1">
         + <EntityType Name="WorkflowTask" sap:content-version="1">
         + <EntityType Name="HeaderDetail" sap:content-version="1">
         + <EntityType Name="ItemDetail" sap:content-version="1">
         + <EntityType Name="UserDetails" m:HasStream="true" sap:content-version="1">
         + <EntityType Name="SupplierDetail" m:HasStream="true" sap:content-version="1">
         + <EntityType Name="ServiceLine" sap:content-version="1">
         + <EntityType Name="Limit" sap:content-version="1">
         + <EntityType Name="PricingCondition" sap:content-version="1">
         + <EntityType Name="ForwardingAgent" m:HasStream="true" sap:content-version="1">
         + <EntityType Name="Attachment" m:HasStream="true" sap:content-version="1">
```
SAP NetWeaver Gateway

OData In Practice – Retrieving Data from an OData Service
Retrieving Data from an OData service

To start with, we will cover the simplest two operations that can be performed using OData: ReadEntitySet and ReadEntity (also referred to as QUERY and READ).

- **ReadEntitySet (QUERY)**: Returns 0..n entries from a collection
- **ReadEntity (READ)**: Returns 0..1 entries from a collection

Both operations are read-only and therefore use the HTTP GET method.

<table>
<thead>
<tr>
<th>REST Operation</th>
<th>HTTP Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a resource</td>
<td>POST</td>
</tr>
<tr>
<td><strong>Retrieve one or more resources</strong></td>
<td>GET</td>
</tr>
<tr>
<td>Update a resource</td>
<td>PUT</td>
</tr>
<tr>
<td>Delete a resource</td>
<td>DELETE</td>
</tr>
</tbody>
</table>
Retrieving a Collection – Constructing the URL

The URL can be constructed with the help of the Service Document:

```xml
<app:service
 xmlns:app="http://www.w3.org/2007/app"
 xmlns:atom="http://www.w3.org/2005/Atom"
 xmlns:sap="http://www.sap.com/Protocols/SAPData"
 xml:lang="en">
  <app:workspace>
    <atom:title type="text">Data</atom:title>
    <app:collection
      sap:creatable="false" sap:updatable="false" sap:deletable="false",
      sap:addressable="false" sap:content-version="1" href="WorkflowTaskCollection">
      <atom:title type="text">WorkflowTaskCollection</atom:title>
      <sap:member-title>WorkflowTask</sap:member-title>
    </app:collection>
  </app:workspace>
  <atom:link rel="self" href="[...]"/>
  <atom:link rel="latest-version" href="[...]"/>
</app:service>
```

Retrieving a Collection – Understanding the Results

Issuing the previously constructed URL from the browser’s address line will cause the Gateway server to return an OData XML message containing the requested collection.

```xml
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://schemas.microsoft.com/mapi/"

  ...  
  <link href="WorkflowTaskCollection" rel="self" title="WorkflowTaskCollection"
  <entry>
  <link href="WorkflowTaskCollection('000001106057')" rel="self" title="WorkflowTaskCollection('000001106057')/
  <link href="WorkflowTaskCollection('000001106057')/HeaderDetails" title="WorkflowTaskCollection('000001106057')/HeaderDetails"
  <content type="application/xml">
  <m:properties>
   <d:WorkitemID>000001106057</d:WorkitemID>
   <d:TaskType>TS20000166</d:TaskType>

  ... 
   </m:properties>
  </content>
  </entry>
</feed>
```
Retrieving a Collection – Filtering

You can filter the <feed> contents by using the $filter query string parameter.

The $filter parameter can contain a complex filter structure consisting of logical, arithmetic, and grouping operators. Also available are string, date, math, and type-related functions.

/WorkflowTaskCollection?$filter=Value gt 2000 and Currency eq 'USD' would only return Workflow Tasks whose PO's total value exceeded $2000 USD.

**IMPORTANT!**
1. You must implement the ABAP coding that responds to this parameter
2. A space character (or %20) **must** be included on either side of the operator in the $filter condition.
3. Enclose non-numeric values within single quotes.
Retrieving a Collection – Paging

If a collection has a high number of entries, then you should use the $top and $skip parameters together to implement paging.

$top specifies the maximum number of entries that should be returned; $skip specifies how many entries should be ignored when selecting the results.

WorkflowTaskCollection?$top=5$ would reduce the <feed> down to the first 5 entries.

WorkflowTaskCollection?$skip=5$ would omit the first 5 entries from the <feed>.

Paging is achieved by setting $top to the number of entries per page, and then incrementing $skip by this number every time the next page is required.
Retrieving a Collection – Expanding

When requesting data for an entity type that is associated with other entity types, the Gateway server will provide you with a navigation URL from which you can retrieve the associated data.

If you are certain that you will also need data coming from associated entity types, you can request associated data in advance by using the $expand parameter. The Gateway server will then include the data from the specified associations in the feed entries.

/WorkflowTaskCollection?$expand=HeaderDetails will return a feed of purchase orders, extended with each Purchase Order’s “header details”

/WorkflowTaskCollection?$expand=HeaderDetails,HeaderDetails/ItemDetails will return a feed of purchase orders with header details, in turn extended with all the items on every PO.
Retrieving a Single Entry

OData employs a special resource path syntax to reference a single entity from a collection. Essentially, you specify the collection name followed by all keys and their respective values that identify the entity in question. Multiple keys are separated with a comma.

/WorkflowTaskCollection(WorkitemID='000001105834') will return a single <entry> from the workflow task collection with the specified ID.

You will also need this special resource path syntax when navigating between associated entities. However, the appropriate syntax is already applied in an <entry>'s <link> elements:

```
<entry>
  <link href="WorkflowTaskCollection('00001106057')" rel="self" title="WorkflowTaskCollection('00001106057')/HeaderDetails" title="WorkflowTaskCollection('00001106057')/HeaderDetails">
    ...  
  </link>
  <content type="application/xml">
    ...
  </content>
</entry>
```
Retrieving a Single Value

Based on an entity read, you can limit your request so it only returns a single property. The OData service will then return the property’s value in plain text.

This can be achieved using the $value navigation:
Based on the resource path of a single entry, append the property’s name as well as $value as if they were navigation paths.

/WorkflowTaskCollection('000001106057')/SupplierName/$value would only return the Purchase Order’s supplier name, not the entire Purchase Order.

In order to limit the resulting entries to several properties, you may use the $select parameter. $select should contain a comma-separated list of property names that are to be retrieved.

/WorkflowTaskCollection?$select=Value,Currency would only list total value and currency for every Purchase Order.
No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

National product specifications may vary.

These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries. Please see http://www.sap.com/corporate-en/legal/copyright/index.epx#trademark for additional trademark information and notices.