3

OData und das SAP Gateway 2.0

3.1 Überblick OData 3-3

3.2 REST 3-5

3.3 OData 3-6

3.4 OData-Webservice 3-7

3.5 CRUD 3-11

3.6 Struktur von OData-Services 3-13

3.6.1 Elemente 3-13

3.6.2 Datenmodell 3-14

3.6.3 Dokumente 3-14

3.7 Überblick SAP Gateway 2.0 3-15

3.8 Komponenten 3-16

3.9 Aufbauszenarien 3-18

3.10 Der SAP Gateway Service Builder 3-21

3.11 SOAP versus OData 3-24

3.12 Anwendungen von OData bei der SAP AG 3-24

3.13 OData in Microsoft Excel & Co 3-24

3.14 OData-Webservices und SAPUI5 3-26

# OData und das SAP Gateway 2.0

## Überblick OData

Fragte man in den vergangenen 25 Jahren einen IT-Verantwortlichen eines beliebigen mittelständischen Unternehmens nach den aktuellen Infrastruktur-Trends, so war die Antwort auf diese Fragestellung sehr oft die gleiche: Einbindung mobiler Endgeräte. Was man unter solchen Geräten verstand, veränderte sich im Laufe der Zeit immer wieder. Egal ob es sich um Notebooks, PDAs, Subnotebook, Netbooks, Smartphones oder Tablets handelte, die infrastrukturellen Problemstellungen blieben immer die gleichen. Wie integrieren wir die Geräte in unsere Infrastruktur? Wie managen wir Soft- und Hardware? Wie stellen wir Daten bereit? Wie stellen wir den Schutz unserer Unternehmensdaten sicher?

Die genannten Fragen stehen exemplarisch für die Probleme, welche sich durch den steigenden Funktionsumfang dieser Geräte immer wieder ergaben.

Im folgenden Abschnitt wollen wir eine Sensibilität für die neusten Entwicklungen im Bereich der Anforderungen an die Integration von mobilen Endgeräten schaffen.

Die explosionsartige Entwicklung der Märkte für Smartphones (seit 2007) und Tablets (seit 2010) transformierte in den vergangenen Jahren die gesamte IT-Landschaft. Während das Wachstum im Bereich der Desktop PCs zwischenzeitlich fast vollständig zum Erliegen kam, drängten immer weitere Geräteklassen auf den Markt. Im Ergebnis sehen wir heute eine sehr große Anzahl unterschiedlichster Endgeräte und Plattformen, von denen jedoch nur wenige eine dominierende Rolle erlangen konnten. Die Wenigen kennzeichneten sich durch sehr gute Integration in das eigene Ökosystem, ständige Verfügbarkeit der benötigten Informationen, einfach zu erlernende Bedienkonzepte und die Rückbesinnung auf wesentliche Funktionen. Die Erfahrungen, welche Anwender jener Endgeräte erhielten, sind nun die Basis der steigenden Erwartungshaltung der Anwender in Unternehmen. Aus dieser Erwartungshaltung ergeben sich neue, fundamentale Anforderungen für die Interoperabilität, welche beispielsweise in fortwährenden Bestrebungen zur Standardisierung von Schnittstellen, Protokollen und Austauschformaten führen.

Projiziert man die bereits genannten Anforderungen nun in   
IT-Infrastrukturen von Unternehmen, so stehen zwei wesentliche Aspekte im Zentrum der Betrachtung:

* Die Integration mobiler Lösungen mit bestehenden Softwareprodukten, wie ERP oder Datawarehouse-Systemen.
* Die Gewährleistung von Datenschutz und Datensicherheit in Bezug auf Unternehmensdaten, welche mit mobilen Endgeräten aus dem Unternehmensstandort hinausgetragen werden und in den Händen von Unbefugten erheblichen Schaden nach sich ziehen könnten.

Diese Aspekte sollen daher in den folgenden Abschnitten genauer betrachtet und mögliche Lösungsansätze aufgezeigt werden.

## REST

Mit REST wird ein Paradigma basierend auf 6 Grundprinzipien beschrieben, welche das Verhalten zahlreichen Beteiligten in einem vernetzten Softwaresystem regeln soll wobei das WWW selbst das Musterbeispiel darstellt. Jedoch ist das Transportprotokoll HTTP selbst nicht zwingend RESTful da mittels diesem auch SOAP genutzt werden kann welches nicht mit den Prinzipen von REST vereinbar ist.

Die sechs Grundprinzipien von REST lauten:

* **Client-Server-Architektur**

Die Beteiligung an einer REST-basierten Applikation muss in einem Client-Server-Aufbau abbildbar sein, um Logikimplementierung auf beiden Seiten voneinander unabhängig zu machen. Es wird davon ausgegangen das auf Seiten des Client keine langfristige Datenpersistenz erfolgt während der Server keinerlei Ambitionen zur Darstellungsweise der Daten hegt welche er liefert. Damit wird eine Trennung der Interessen erreicht.

* **Zustandslosigkeit**

Bei jeder Kontaktaufnahme durch den Client muss dieser sämtliche Informationen mitliefern die zur Bearbeitung durch den Server notwendig sind auch wenn sich diese seit der letzten Verbindung nicht verändert haben. Auf diese Weise wird den Konsequenzen aus verteilten Serversystemen Rechnung getragen.

* **Zwischenspeicherbarkeit**

Um die Leistung und Verfügbarkeit zu verbessern ist jeder Datenaustausch zum Client mit einem Kennzeichen zu versehen, ob der Empfänger die erhaltenen Informationen zwischenspeichern darf. Hierhinter steckt der Gedanke, dass der Client nicht mit einem veralteten Datenbestand arbeiten soll.

* **Schichtenaufbau**

Für den Client darf es nicht ersichtlich sein, ob er eine Anfrage direkt an den zuständigen Server stellt oder ob diese Anfrage aus Leistungs- oder Sicherheitsgründen von einem vorgeschalteten System verarbeitet wird.

* **Einheitliche Schnittstelle**

Um die Entkopplung von Clients und Server zu verbessern, muss die einheitliche Schnittstelle folgende Eigenschaften aufweisen:

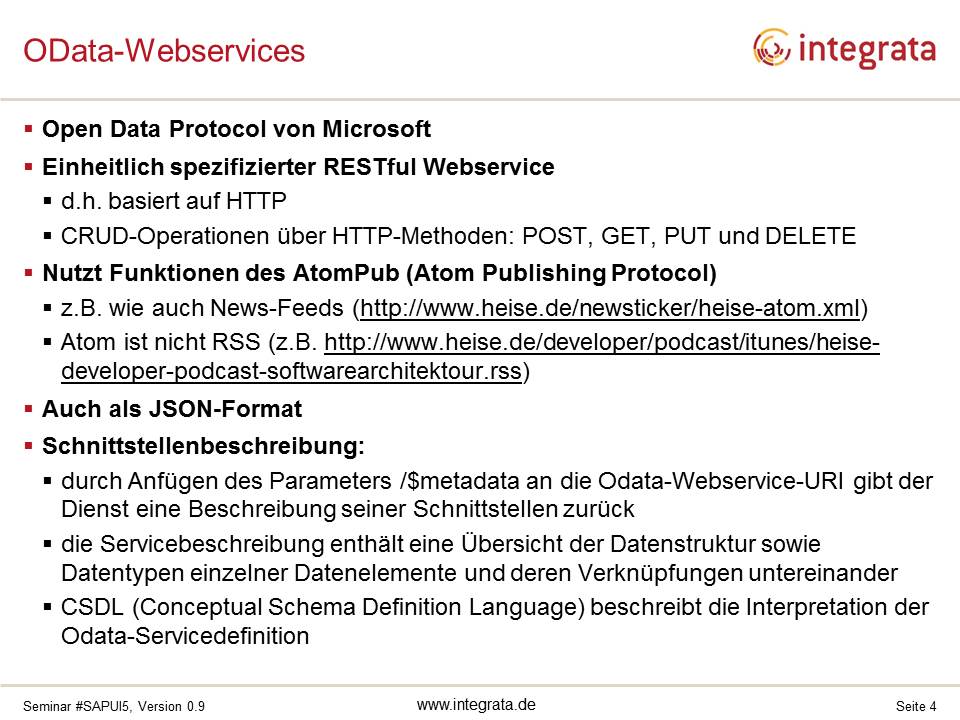
* Der Server muss dem Client ansprechbare Adressen für seine Ressourcen zur Verfügung stellen.
* Dem Client müssen Wege zur Verfügung gestellt werden diese Ressourcen zu nutzen (bspw. lesen, beschreiben, erstellen, löschen).
* Diese Manipulationen durch den Client müssen hypermedia-basiert erfolgen.
* Alle Serverantworten müssen selbsterklärend sein.
* **Ausführung auf Abruf**

Der Server muss auf Anfrage dem Client Ausführungslogik zur Verfügung stellen. Bei OData wird dies durch die Verwendung von Function Imports ermöglicht.

## OData

OData entstand 2007 durch Microsofts bestreben eine standardisierte, leistungsstarke Datenschnittstelle zwischen serverseitig betriebenen Geschäftsanwendungen und ihren lokal ausgeführten Benutzeroberflächen zu schaffen. Während die Spezifikationen der Versionen 1 bis 3 dabei noch durch Microsofts Open Specification Promise – einer öffentlichen Selbstverpflichtung Microsofts – veröffentlicht wurden, gelangte es 2012 in die Obhut der OASIS welche die Spezifikation der Version 4.0 unter Mitwirkung zahlreicher Unternehmen, darunter auch SAP, erarbeitete und im darauffolgenden Jahr veröffentlichte.

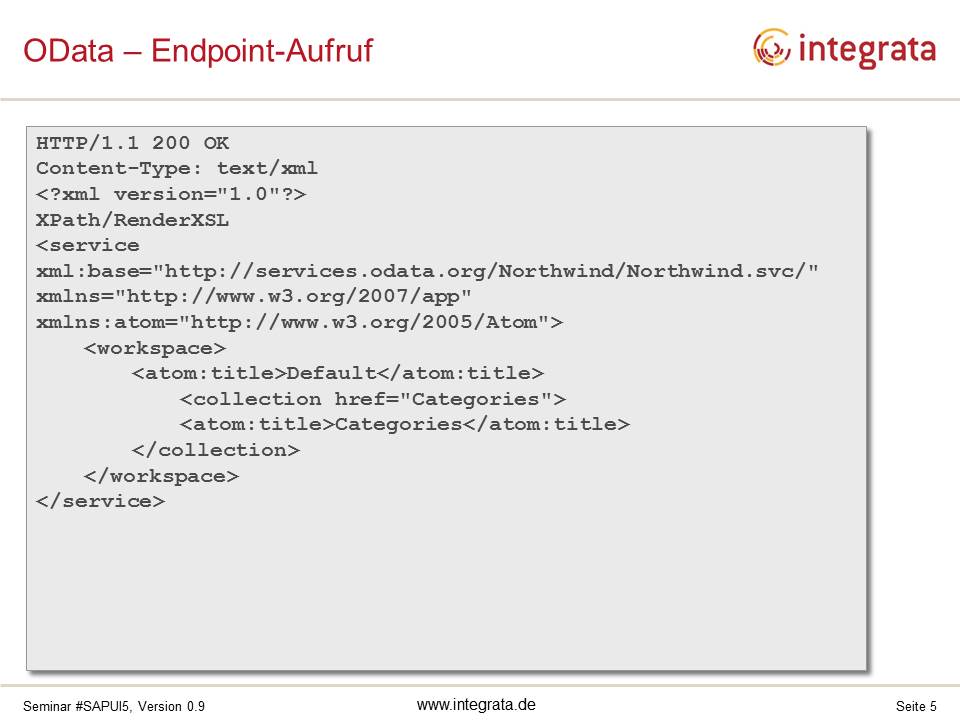
## OData-Webservice



Da sich OData auf HTTP ist jeder Webbrowser dazu in der Lage mit ihm bereits auf Entdeckungsreise zu gehen. Gegenwärtig sind die Anfragen und Antworten bei OData auf AtomXML bzw. JSON standardisiert wobei letzteres

* Einen grundsätzlich geringeren Protokolloverhead erzeugt und damit die zu übertragende Datenmenge erheblich verringert
* bei OData allmählich AtomXML als Standardformat ablöst.

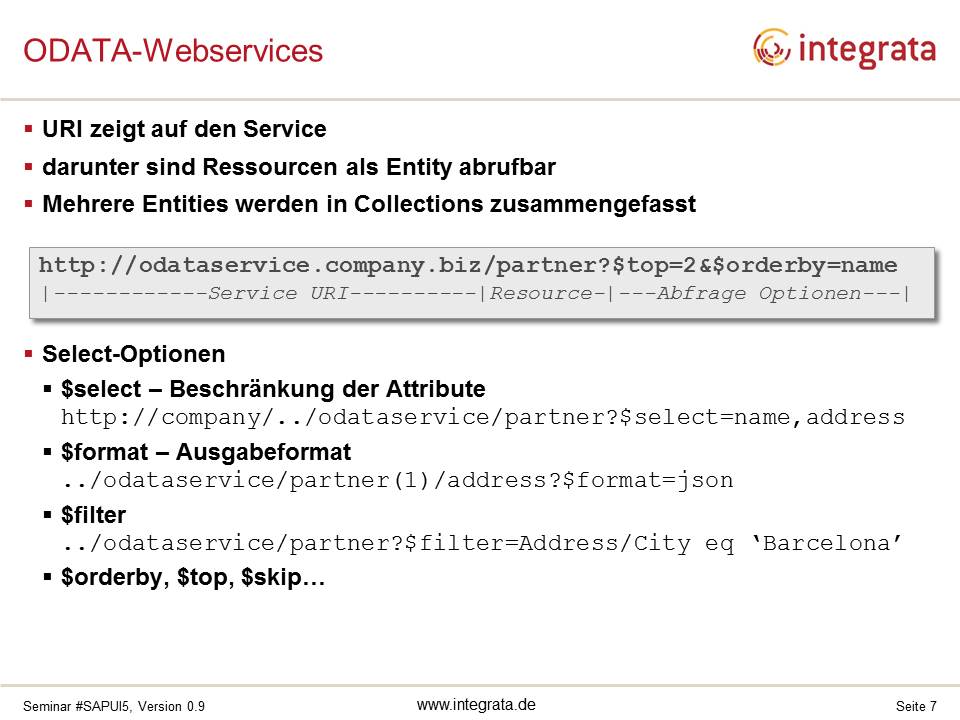
Bei der Betrachtung eines OData-Service direkt im Browser ist zu einer Browsererweiterung zu raten welche die Darstellung für den Menschen verbessern. Mittels des Schalters $format kann dabei zwischen AtomXML ($format=xml) und JSON ($format=json) gewechselt werden. Im Falle des SAP Gateway als Backend kann out-of-the-box eine bessere Darstellung die Ergänzung der URL um den Parameter $sap-ds-debug=true erreicht werden.



Einem OData-Service liegt neben dem Servicedokument und dem Metadatendokument das eigentliche logische Datenmodell zugrunde. Beim direkten Aufruf des Services beantwortet der Server mit dem Servicedokument in welchem der allgemeine Aufbau (Entitätscontainer) des Datenmodells dargestellt wird. Ergänzt man diesen direkten Aufruf um den Parameter $metadata erhält man das Metadatendokument mit der genaueren Beschreibung:

* innerer Aufbau des Entitätscontainers
* dessen Entitäten
* Eigenschaften der Entitäten (Schlüsselfeld, Datentypen)
* Beziehung zwischen den Entitätscontainer und Navigationsmöglichkeiten durch diese Beziehungen.









## CRUD

Auf OData-Webservices, genauer auf ihre Entitäten/Ressourcen, sind dabei die folgenden vier Operationen möglich:

* Datensatz anlegen (**C**reate; HTTP-Methode POST)
* Datensatz/-sätze lesen (**R**ead; HTTP- Methode GET)
* Datensatz aktualisieren (**U**pdate; HTTP- Methode PUT)
* Datensatz löschen (**D**elete; HTTP- Methode DELETE)



Um sämtliche Datensätze einer Entität abzurufen ist die Angabe der Service URI sowie der Ressource ausreichend, lässt sich jedoch mit den bekannten Abfrageoptionen einschränken.

Der Abruf eines einzelnen Datensatzes erfordert dagegen die Angabe des eindeutigen Schlüssels unmittelbar im Anschluss an die Ressource.



Es versteht sich das, mit Ausnahme von $expand, sämtliche Abfrageoptionen unnötig sind.

Ganz ähnlich erfolgt sowohl das Löschen als auch das Ändern eines bestimmten Datensatzes mit dem winzigen Unterschied in der anzuwendenden HTTP-Methode. Möchte man mehrere Datensätze mit einem Aufruf löschen bzw. ändern so sei auf die Abfrageoption $batch verwiesen auf welche später eingegangen wird.

Zur Abrundung erfolgt das Anlegen eines neuen Datensatzes wieder ausschließlich die Verwendung der Service-URI samt Ressource unter Nutzung der HTTP-Methode POST. Während die beiden anderen Daten-manipulierenden Operationen aber nur eine HTTP-Statusantwort liefern antwortet der OData-Webservice hier bei erfolgreichem Abschluss jedoch zusätzlich mit dem vollständigen Datensatz. Dies ist dem Umstand geschuldet das so manches Element des Satzes erst vom Backend belegt wird wie bspw. eine eindeutige ID oder einem Zeitstempel.

**Batch-Processing**

Mittels der Abfrageoption $batch ist es möglich mehrere Abfragen zu einer einzigen zu bündeln. Unter Verwendung des Content-Type multipart und der HTTP-Methode POST kann der Webservice mehrere Operationen ausführen und beantwortet sie auch einmalig. Hierbei sind jedoch Einschränkungen zu beachten:

1. Ein Batch-Prozess kann entweder nur Lese- oder ausschließlich Schreiboperationen enthalten, denn
2. Schreibende Batch-Prozesse können nur vollständig oder gar nicht abgearbeitet werden

Aus letzterem Grunde ist deshalb strengstens auf referentielle Integrität der Operationen zu achten.

**ETags**

Um die zu übertragende Datenmenge weiter zu reduzieren, bietet OData die grundsätzliche Unterstützung von ETags. Ist diese Funktionen im Webservice implementiert so kann der dieser ab der zweiten, lesenden Abfrage einer Ressource das Datendelta ermitteln und die Übertragung auf dieses beschränken. Darüber hinaus kann dieses Feature genutzt werden, um die Plausibilität von Schreiboperationen zu überprüfen: Bevor ein Datensatz gelöscht oder geändert werden soll wird der Service befragt ob dieser Satz seit seiner Übermittlung eine Manipulation erfahren hat. Der Auftraggeber der Schreiboperation kann dann entscheiden, ob er seine Schreiboperation dennoch ausgeführt werden soll oder nicht.

Des Weiteren sind ETags ein Baustein bei der Implementierung von Notifications/OData-Push wobei der Webservice in die Lage versetzt wird den Nutzer aktiv mit neuen oder geänderten Daten zu versorgen.

## Struktur von OData-Services

Ein OData-Service ist ein Web Service, welcher Zugriff auf Ressourcen (Daten) anbietet und für diesen Zugriff das OData-Protokoll implementiert.

Da bei einem OData-Service die Ressourcen im Mittelpunkt stehen, ist auch seine Struktur durch die Beschreibung der angebotenen Daten bestimmt.

In diesem Kapitel werden die Elemente vorgestellt, aus denen die Struktur aufgebaut ist und die Dokumente erklärt, in denen diese Struktur publiziert wird.

### Elemente

#### Entität

Eine Entität entspricht einem Datensatz einer Ressource, vergleichbar mit einer Instanz in der Objektorientierung, welche verschiedene Eigenschaften besitzt. Welche Eigenschaften und welche Form diese haben ist durch den Entitätstyp definiert, außerdem sind einige als Schlüssel definiert, jede Entität muss eindeutig über diesen Schlüssel identifiziert werden können.

#### Entitätstyp (Entity Type)

Wenn der Vergleich der objektorientierten Programmierung fortgeführt wird, entspricht ein Entitätstyp einer Klasse. Er definiert, welche Eigenschaften von welchem Datentyp möglich sind, über welche Schlüsselattribute eine Entität identifiziert werden soll und welche Navigationseigenschaften zu anderen Entitätstypen existieren.

#### Entitätsmenge (Entity Set)

Mit Entitätsmengen werden die zur Laufzeit vorhandenen Entitäten verwaltet. Jede Entitätsmenge basiert auf einem Entitätstyp, es können aber mehrere Entitätsmengen auf dem gleichen Typ basieren.

#### Attribut (Property)

Attribute oder Eigenschaften haben viel Ähnlichkeit mit den Feldern einer Datenbanktabelle, sie können Daten von einem elementaren Datentypen oder strukturierte Daten beinhalten, die Typisierung eines Attributs erfolgt einmalig im Entitätstyp.

#### Navigationseigenschaft (Navigation Property)

Eine Navigationseigenschaft ist ein spezielles Attribut, welches die Verknüpfung (Link) eines Entitätstyps mit einem zweiten darstellt. Die Kardinalität einer Navigationseigenschaft bestimmt, ob eine Entität mit genau einer anderen oder einer Entitätsmenge verbunden ist. Navigationseigenschaften repräsentieren die praktische Umsetzung einer Assoziation.

#### Assoziation

Durch eine Assoziation wird die Beziehung von zwei Entitätstypen zueinander dargestellt. Hierzu werden in der Assoziation die beiden verknüpften Entitätstypen, ihre Kardinalitäten und die Attribute definiert, über die die Relation hergestellt wird (Schlüssel-Fremdschlüssel-beziehung).

### Datenmodell

Die Menge aller dieser Elemente eines Services wird Datenmodell genannt. In einem späteren Kapitel wird auf das Anlegen eines solchen Datenmodells mitsamt den oben beschriebenen Elementen näher eingegangen und mit einem Beispiel illustriert.

### Dokumente

Im Folgenden werden die beiden Dokumente vorgestellt, über die der OData-Service seine Struktur nach außen hin bekannt gibt. Diese Bekanntgabe ist notwendig, möglichen Nutzern die verfügbaren Ressourcen, Funktionen und Relationen des Service aufzuzeigen.

#### Servicedokument

Das Servicedokument eines OData-Services erhält man, wenn man die Wurzel-URI des Services aufruft, wo es im XML oder JSON Format verfügbar ist.

Zweck des Dokuments ist es alle Entitätsmengen des Service aufzulisten.  
Jede Entitätsmenge ist durch einen <app:collection>-Tag gekennzeichnet, in welchem Sie den jeweiligen Namen der Entitätsmenge (welcher gleichzeitig den Pfad zu dieser Menge relativ von der Wurzel-URI darstellt) und den zugrunde liegenden Entitätstyp.

**Hinweis:** Für Services, die aus einem SAP-System bereitgestellt werden, können Sie den Abfrageparameter sap-ds-debug=true verwenden, um eine aufbereitete HTML-Ansicht zu erhalten. Besonders praktisch sind die durch Klicks navigierbaren Links

#### Service-Metadatendokument

Das Metadatendokument eines Services enthält sämtliche Informationen des Datenmodells, also Entitätstypen, Relationen und Navigationseigenschaften und Entitätstypen.

Sie erhalten dieses Dokument, indem Sie $metadata an die Wurzel-URI anhängen. Dieses Dokument ist dazu dar den Konsumenten des Services über den Aufbau und mögliche Operationen des Services zu informieren.

Zum Teil kann dieses Dokument von Entwicklungswerkzeugen genutzt werden, um Zugriffsmethoden auf den Service automatisch zu implementieren.

## Überblick SAP Gateway 2.0



## Komponenten



Bis zur Version 7.31 des SAP NetWeaver Application Server für ABAP (AS ABAP 7.31) setzt sich das SAP Gateway aus mehreren Komponenten zusammen:

* **GW\_CORE & IW\_FND**

Als Kernkomponente des SAP Gateways.

* **IW\_BEP** (Business Enablement Provisioning)

Erlaubt die Erzeugung von OData Services auf Basis von BAPIs, remotefähigen Funktionsbausteinen, Business Workflows sowie Reportingobjekten des SAP Business Warehouses anhand von MDX bzw. Easy Queries.

* **IW\_PGW** (Process Gateway)

Bereitstellung von SAP Business Workflows und SAP BPM als ODataService.

* **IW\_SPI** (Service Provider Infrastructure)

Notwendig zur Nutzung von User Interface Building Blocks (UIBB), Personal Object Worklist-Komponenten (POWL), Adobe Forms sowie zu Integration des Floor Plan Managers oder sämtlicher anderer Inhalte, welche das Service Provider Interface nutzen.

* **IW\_HDB**

Zur Darstellung von OData-Services basierend auf SAP HANA-Artefakten wie Attribute oder Analytic Views.

* **IW\_CNT & IW CBS**

Vordefinierte Inhalte aus der Business Suite als Basis für OData-Services.

* **IW\_SCS** (Screen Scraping)

Erzeugung von OData-Services basierend auf Inhalten des Screen ScrapingGenerators.

* **IW\_GIL** (Generic Interaction Layer)

Erzeugung von OData-Services basierend auf der Verbindungsschicht zwischen Business Object Layer (BOL) und den Datenbanktabellen.

Diese Komponenten sind dabei im Aufbauszenario des Central Hub auf die beteiligten Systeme zu verteilen.

Mit dem Release 7.40 des AS ABAP wurde das SAP Gateway zur Kernkomponente, sodass es standardmäßig bereits mit installiert wird. Dabei erfolgte auch eine Konsolidierung der Komponenten wobei GW\_CORE, IW\_FND, IW\_BEP und IW\_HDB zur Basiskomponente SAP\_GWFND miteinander verschmolzen. Sämtliche anderen Komponenten sind entsprechend des Aufbauszenarios und Bedürfnissen zu installieren.

Durch diesen Schritt kann sich der Aufbau in einer Landschaft welche verschiedene Versionen des AS ABAP aufweist als anspruchsvoll erweisen. Zum Mapping der Versionen vor und ab dem AS ABAP 7.40 konsultieren Sie bitte die SAP Note **1942072**.

## Aufbauszenarien

Es stehen für den Aufbau eines SAP Gateway Systems zwei grundlegend verschiedene Ansätze zur Verfügung: Als so genanntes Central Hub oder Embedded (eingebettet). Während beim eingebetteten Ansatz das SAP Gateway als zusätzliche Softwarekomponente auf ein bestehendes AS ABAP-System installiert (bis Release 7.40) und eingesetzt (ab einschl. AS ABAP 7.40) wird ist beim Central Hub ein dediziertes System notwendig an welches ein oder mehrere Backendsysteme (ERP, SCM usw.) angebunden wird.







Weiterhin wird beim Hub-Ansatz noch dahingehend unterschieden auf welchem System die Entwicklung selbst erfolgt:

1. In einem System der Business Suite
2. Im System des SAP Gateway wobei die Business Suite nicht angefasst werden muss.

In allen Fällen stellt das SAP Gateway den Service Endpoint für die Nutzung dar.

Verfolgt man den Embedded-Ansatz so sollten zusätzliche Sicherheitsmaßnahmen ergriffen werden sollten die Anwendung vom Internet aus erreichbar sein.

## Der SAP Gateway Service Builder

Die Heimstätte zur Gestaltung eines OData-konformen Gateway-Service ist der SAP Gateway Service Builder. Zwar bietet er dem Entwickler jederzeit auch die Möglichkeit zur manuellen Implementierung von Diensten und Funktionen, seine vielfältigen Assistenten sind jedoch eine starke Arbeitserleichterung einen OData-Service zügig zu entwickeln oder zu erweitern.

Der SAP Gateway Service Builder arbeitet dabei Projektbasiert und ist über die Transaktion SEGW jenes Systems aufzurufen, auf welchem die Entwicklung erfolgen soll (vgl. Deployment Scenarios). Ein solches Projekt stellt dabei einen einzigen, in sich abgeschlossenen OData-Service dar welcher für die Zuordnung zum Paket (vgl. SAP Transportwesen) zuständig ist und in verschiedenen Versionen vorliegen kann.

Es ist stark zu empfehlen das eigentliche Datenmodell in der folgenden Reihenfolge einzurichten:

1. Entitätstypen
2. Komplexe Typen
3. Assoziationen (Navigation)
4. Entitätsmengen
5. Assoziationsmengen
6. Function Imports

Bei den **Entitätstypen** handelt es sich um das Analogon zur Dictionary-Struktur - wobei es sich auch direkt um eine solche handeln kann, dazu später aber mehr - und bei den Attributen des Entitätstyps um die Komponenten der Struktur. Zwar kann ein Attribut jedweden ABAP-Datentyp, oder einen komplexen Typ, aufweisen welcher jedoch auf einen der folgenden, primitiven Datentypen abbildbar sein muss:

|  |  |  |
| --- | --- | --- |
| Edm.Binary | Edm.Decimal | Edm.Int64 |
| Edm.Boolean | Edm.Double | Edm.SByte |
| Edm.Byte | Edm.Guid | Edm.Single |
| Edm.DateTime | Edm.Int16 | Edm.String |
| Edm.DateTimeOffset | Edm.Int32 | Edm.Time |

Zu bemerken ist an dieser Stelle die Abwesenheit eines reinen Datentyps für das Datum was eine adäquate Behandlung der Zeitzone im primitiven Datentyp Edm.DateTime erfordert. Zu beachten ist in der JSON-Darstellung die Verwendung der Anzahl der Millisekunden seit Mitternacht des 1. Januars 1970 (UTC).

Eine **Entitätsmenge** wiederum entspricht einer Menge in der Struktur seines Entitätstyps.

**Assoziationen** dienen der logischen Verknüpfung zweier Entitäten/-mengen wie bspw. von Kopf-Entität und Positions-Entitätsmenge. Mittels konfigurierten Navigationsattributs ist es so möglich sich zu einem Kopfbeleg die zugehörigen Positionen über OData ausgeben zu lassen.



Nach erfolgter Modellierung der Daten schließt sich die Serviceimplementierung an wobei zunächst die Serviceartefakte, die Model & Data Provider-Klassen zu generieren sind. Nun sind zwei Ansätze zu unterscheiden:

* Mapping-Ansatz
* Programmatischer Ansatz

**Mapping-Ansatz**

Hierbei werden die notwendigen Methoden (Get, Create etc.) jeweils einem (RFC-)Funktionsbaustein, einem (Remote-)BAPI, der Suchhilfe und ab AS ABAP 7.4 auch CDS zugeordnet, welcher bei Nutzung der Methode vom Gateway aufgerufen wird.

**Programmatischer Ansatz**

Hierbei erfolgt die Serviceimplementierung als Redefinition der jeweiligen Methoden in der erweiterten Data-Provider-Klasse   
(ZCL\_<Projektname>\_DPC\_EXT).

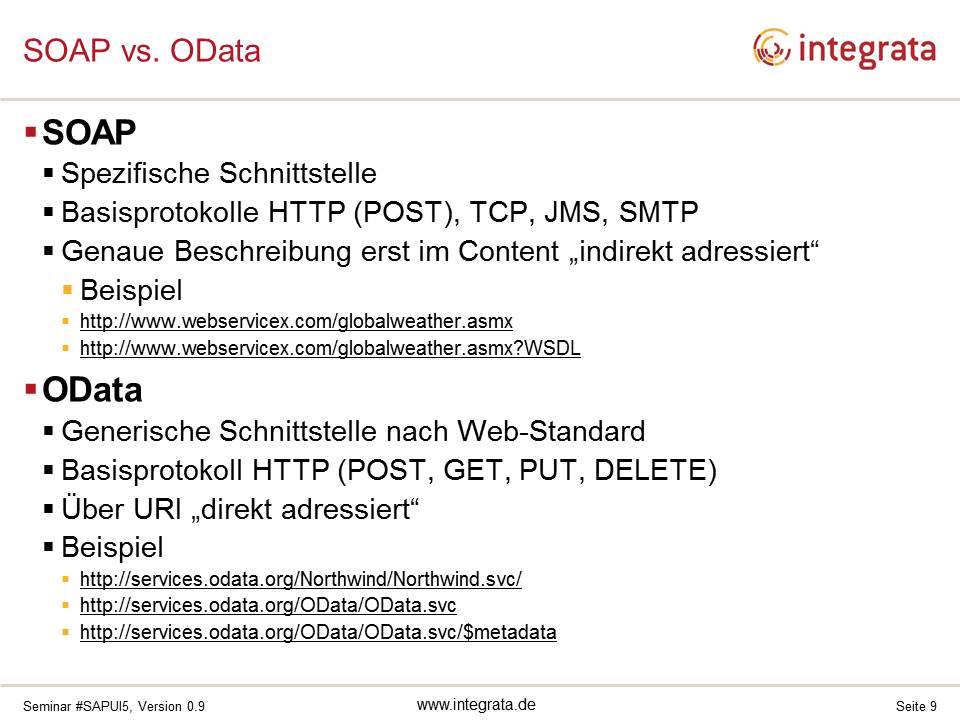
Es besteht die Möglichkeit diese Ansätze auch pro Entität zu mischen jedoch pro Methode einer Entität ist sich auf einen Ansatz festzulegen.

Anschließend ist der Service im Katalog über die Transaktion /IWFND/MAINT\_SERVICE zur registrieren/aktivieren.

Bei diesem Vorgang ist darauf zu achten das im Deployment Scenario des Central Hub mit Entwicklung im Backend der OData-Service in jenem Backend mit dem Systemalias LOCAL und anschließend im Central Hub mit dem Systemalias des Backends zu registrieren und aktivieren ist.



## SOAP versus OData



## Anwendungen von OData bei der SAP AG

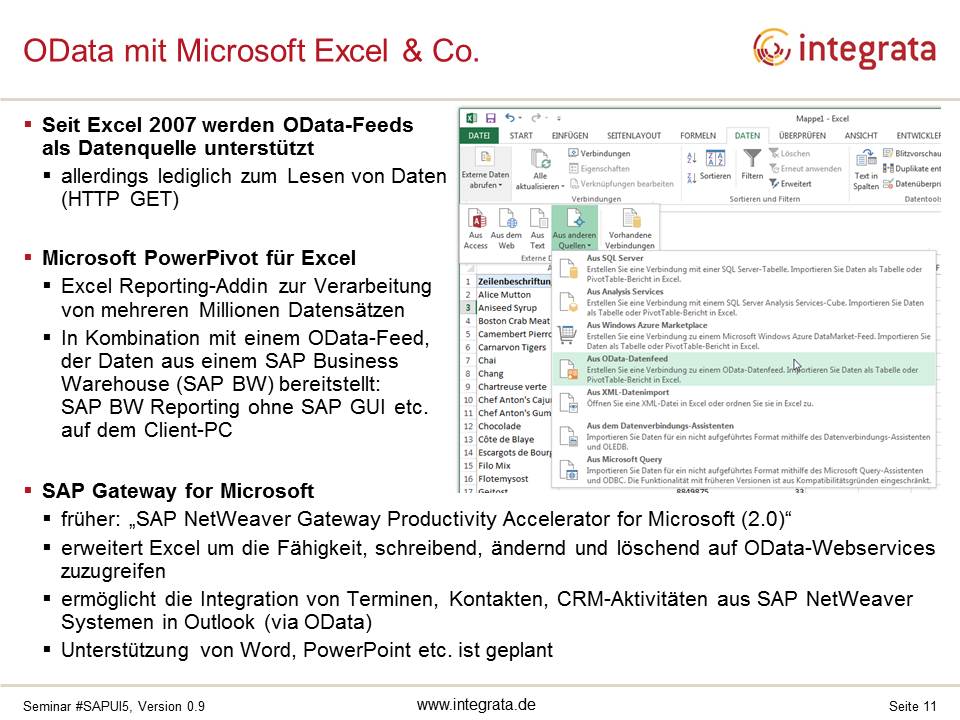


## OData in Microsoft Excel & Co

Bereits seit Excel 2007 unterstützt die Microsoft Tabellenkalkulation OData-Feeds, d.h. OData-Webservices, als externe Datenquelle. Somit können Sie beispielsweise auch Daten eines SAP Business Warehouse in Excel konsumieren, die via SAP NetWeaver Gateway OData-Webservice bereitgestellt werden. In Excel könnten Sie diese dann etwa in einer Pivottabelle ausgeben, die dem grundsätzlichen Aufbau des SAP BEx Analyzers entspricht. Letzteres ist Bestandteil des SAP GUI und als Excel-Addin nach wie vor das gängigste Reporting-Werkzeug für Daten des SAP Business Warehouse.

Um große Datenmengen – auch aus einem OData-Feed – in Excel auswerten zu können, greifen Sie am besten auf Microsoft PowerPivot for Excel zurück. Dieses Excel-Addin enthält für Excel 2010 eine separate Engine, die mehrere Millionen Datensätze verarbeiten kann. Für Excel 2010 ist PowerPivot als kostenloses Addin erhältlich. In Excel 2013 ist es lediglich in Microsoft Office 2013 Professional Plus oder den Excel 2013 Standalone-Versionen enthalten.

Beachten Sie in jedem Fall: Excel kann – ob mit oder ohne PowerPivot – OData-Feeds lediglich konsumieren, d.h. ausschließlich Daten lesen. Ein Schreiben oder Löschen von Daten via POST, PUT oder DELETE ist mit Excel-Bordmitteln bislang leider nicht möglich.



Mit dem Microsoft Office-Addin SAP Gateway for Microsoft (früher: “SAP NetWeaver Gateway Productivity Accelerator for Microsoft“) beheben Sie die im vorherigen Abschnitt genannte Einschränkung. Nach Installation dieses Addin können Sie Daten via OData-Webservice in Excel nämlich nicht nur auslesen, sondern auch editieren (PUT) oder hinzufügen (POST).

SAP Gateway for Microsoft integriert sich jedoch nicht nur in Excel, sondern auch in Outlook: So können Sie via OData beispielsweise CRM-Aktivitäten, Zeiterfassungsdaten oder Kontakte aus einem SAP NetWeaver System in Outlook übernehmen. Eine Integration in Microsoft Word, PowerPoint etc. ist geplant.

SAP Gateway for Microsoft ist mit Stand vom April 2014 als Pro-User-Lizenz erhältlich und kann, sofern eine Lizenz erworben wurde, über den SAP Service Marketplace bezogen werden.

## OData-Webservices und SAPUI5

