SAP SuccessFactors Learning Impact Bulletin

Q4 2016: Apache Olingo Upgrade (4.0.0 to 4.2.0)

This document is published on November 2, 2016. What changed: we wanted to suggest an alternative to the has operator: the contains operator.

BACKGROUND

Olingo is a Java library that implements the Open Data Protocol (OData). The community is moving from Olingo 4.0.0 to Olingo 4.2.0 in anticipation of end of support for 4.0.0.

CHANGES TO HOW YOU CALL SAP SUCCESSFACTORS LEARNING WEB SERVICES

Starting in Q4 2016, you will notice that Olingo 4.2.0 has become stricter. It prevents intermixing of data types, it does not allow short names, and it requires a namespace for the **has** operator. We have taken action to reduce the impact of these changes and give you time to transition to the stricter calls.

Intermixing Data Types now Prevented

Olingo 4.0.0 had a bug that allowed you to intermix data types. In Q4 2016 and later, if you intermix data types, your call will error. In the example below, maxPageSize is a string. The first call fails and the second works.

```
.../financialtransactions/v1/FinancialTransactions?$filter=criteria/maxPageSize eq 1
.../financialtransactions/v1/FinancialTransactions?$filter=criteria/maxPageSize eq '1'
```

Short Entity Names Deprecated

Olingo 4.0.0 allowed short entity names but Olingo 4.2.0 requires fully qualified entity names (or it can be removed). We understand that our documents showed the short name so clients were built with the short name. To compensate, we currently intercept the short name to avoid errors, but it is deprecated: we want customers to begin removing the short name. This issue affects insert or update (PUT or POST). GET calls are not affected.

Deprecated

```
POST learning/odatav4/public/admin/financialtransactions/v1/FinancialTransactionPostingStatuses
{"@odata.type":"#FinancialTransactionPostingStatuses","postingStatuses":[…]}
```

Recommended

To name an entity type (as opposed to an entity set):

```
"@odata.type":"#com.sap.lms.odata.User"
```

To name an entity set (as opposed to an entity type):

```
"@odata.context": "$metadata#Users/$entity"
```

Alternative

As an alternative, you can remove the entity name:

```
POST learning/odatav4/public/admin/financialtransactions/v1/FinancialTransactionPostingStatuses
{"postingStatuses":[…]}
```

Removal of has operator
After an investigation of calls into our API layer, we found zero calls with the **has** operator but found that **eq** operator is most commonly used. Olingo 4.2.0 strictly follows the Oasis standard, which means that if we allow **has**, then both **has** and **eq** must use a namespace:

```
$filter=scriteria/learnerID has com.sap.lms.odata.ENString 'thomas'
$filter=scriteria/learnerID has Namespace1_Alias.ENString 'thomas'
$filter=scriteria/learnerID eq com.sap.lms.odata.ENString 'thomas'
$filter=scriteria/learnerID eq Namespace1_Alias.ENString 'thomas'
```

Given that we saw no calls with **has**, and given that its enablement would have meant major refactoring on the most commonly used **eq** call, we disabled **has** to protect the current usage of **eq**. We decided to reduce the impact to your clients. We will continue to work with the Olingo team to find a path forward.

Alternative
As an alternative to the `has` operator, we recommend that you use the `contains` operator. It provides the same fuzzy searching as `has` but is easier to use. For example, you could search for all users whose student ID (learner ID) `contains` thomas.

```
$filter=contains(scriteria/learnerID,'thomas')
```

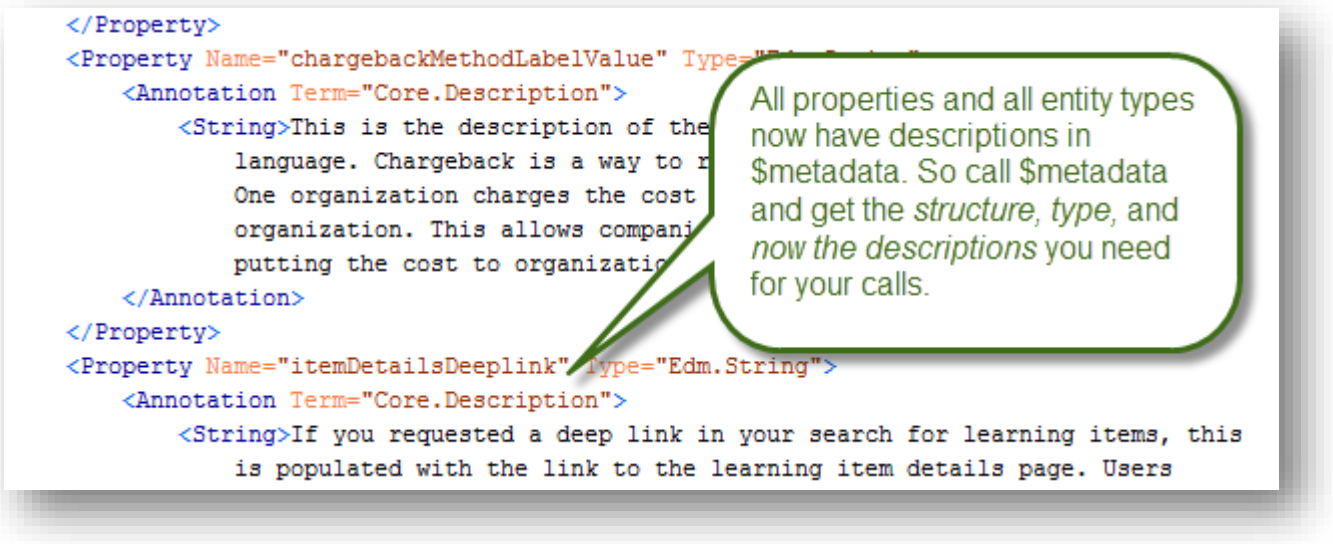
IMPACTS TO ODATA EXTENSIONS

If you have built an SAP SuccessFactors Learning extension that extends the OData library, you must check that extension to make sure that it works properly after the update. Other types of extensions are *not* affected.

CHANGES TO HOW YOU ACCESS REFERENCE DOCUMENTATION

Starting in Q4 2016, you call $metadata to see all reference information for the entities in your call. We have moved to programmatic reference documentation (like JavaDoc or Doxygen) to reduce errors. Because the types, limits, relationships, and names are read from the code itself, they cannot mismatch the code. We have added comments in $metadata so that you can understand the meaning of the fields.

Starting in Q4 2016, $metadata looks like the following:



You can still find a PDF telling you how to call $metadata, the reasons why you would make the calls, how to get a OAuth token, and any other information you need *except for* the reference information.