successfactors™
An SAP Company

OData Query

Release Information

Product Version 1308
Release Date 9/2013
© 2013 SuccessFactors Inc. All rights reserved.

Contact Information

SuccessFactors Global Headquarters
1 Tower Place, Suite 1100
South San Francisco, CA 94080
USA
Toll Free: 800-809-9920
Phone: 650-645-2000
Fax: 650-645-2099

http://www.successfactors.com/

Contents

1 OData Query

OData is a REST based protocol for accessing a data store. The entities in the API are described in the metadata document. Accessing the entities is as simple as knowing the entity name, and then referencing the entity through an HTTP Get. For example, all SuccessFactors systems have a “User” entity which represents a user account. To query for all the users in a system is as simple as accessing the following URL:

https://<hostname>/odata/v2/User

Of course, there are many more things to consider, including filtering, paging, sorting, joining to other entities (known as “expanding” in OData), etc. Examples of these details are described below.

Note: The OData protocol does not dictate that any particular operation must be supported. It allows the service provider (that means SuccessFactors for this API) to decide which operations are supported.

In our system, the supported operations vary by entity. For example, our system does not allow user accounts to be deleted. We allow them to be set as inactive only. Therefore the “User” entity does not support the delete operation.

This functional guide does not cover the details of each entity behavior. SuccessFactors will provide a number of separate entity guides, organized by solution, to describe the entity supported operations, business meaning and business logic, and configurations details like security and custom field configuration.

1.1 Query Operation

This chapter explains how to compose an OData URI, and explains each component of the URL. Afterwards we present some query examples.

In OData, a URI has the following structure:

http://<hostname>/odata/v2/EntitySet[(keyPredicate)][/navPropSingle][/NavPropCollection][/Complex][/Property]

Below is a description of each of the parts of the above URI:

- **EntitySet:** The name of a Collection or a custom Service Operation (which returns a Collection of Entries) exposed by the service.
- **KeyPredicate:** A predicate that identifies the value(s) of the key Properties of an Entry. If the Entry has a single key Property the predicate may include only the value of the key Property. If the key is made up of two or more Properties, then its value must be stated using name/value pairs. More precisely, the syntax for a KeyPredicate is shown by the following figure.

- **NavPropSingle:** The name of a Navigation Property defined by the Entry associated with the prior path segment. The Navigation Property must identify a single entity (that is, have a "to 1" relationship).
- **NavPropCollection**: Same as NavPropSingle except it must identify a Collection of Entries (that is, have a "to many" relationship).
- **ComplexType**: The name of a declared or dynamic Property of the Entry or Complex Type associated with the prior path segment.
- **Property:** The name of a declared or dynamic Property of the Entry or Complex Type associated with the prior path segment.

The example URIs here follow the addressing rules stated above and are based on the reference service and its metadata available

http://<hostname>/odata/v2/User

- Identifies all User Collection.
- Is described by the Entity Set named "User" in the service metadata document.

http://<hostname>/odata/v2/User('1')

- Identifies a single User Entry with key value 1.
- Is described by the Entity Type named "User" in the service metadata document.

http://<hostname>/odata/v2/User('1')/username

- Identifies the Name property of the User Entry with key value 1.
- Is described by the Property named "Name" on the "User" Entity Type in the service metadata document.

http://<hostname>/odata/v2/User('1')/proxy

- Identifies the collection of proxy associated with User Entry with key value 1.
- Is described by the Navigation Property named "proxy" on the "User" Entity Type in the service metadata document.

http://<hostname>/odata/v2/User('1')/proxy/$count

- Identifies the number of proxy Entries associated with User 1.
- Is described by the Navigation Property named "proxy" on the "User" Entity Type in the service metadata document.

http://<hostname>/odata/v2/User('1')/proxy('1')/hr/username

- Identifies the username of the hr for proxy 1 which is associated with User 1.
- Is described by the Property named "username" on the "hr" Entity Type in the service metadata document.

http://<hostname>/odata/v2/User(‘1’)/proxy(‘1’)/hr/username/$value

- Same as the URI above, but identifies the "raw value" of the username property.

1.2 Query Links

Much like the use of links on Web pages, the data model used by OData services supports relationships as a first class construct. For example, an OData service could expose a Collection of proxy Entries each of which are related to a User Entry.

Associations between Entries are addressable in OData just like Entries themselves are (as described above). The basic rules for addressing relationships are shown in the following figure.

http://<hostname>/odata/v2/EntitySet[(keyPredicate)]/$links/navigationProperty

- **NavigationProperty**: The name of a Navigation Property that is declared on the Entry associated with the path segment prior to the "$links" segment.

Examples

The example URIs here follow the addressing rules stated above and are based on the reference service and its metadata available at http://<hostname>/odata/v2/$metadata.

http://<hostname>/odata/v2/User(‘1’)/$links/proxy

- Identifies the set of proxy related to User 1.
- Is described by the Navigation Property named "proxy" on the "User" Entity Type in the associated service metadata document.

http://<hostname>/odata/v2/User(‘1’)/$links/proxy

- Identifies the proxy related to User 1.
- Is described by the Navigation Property named "proxy" on the "User" Entity Type in the associated service metadata document.

1.3 Query String Options

1.3.1 System Query Option

System Query Options are query string parameters a client may specify to control the amount and order of the data that an OData service returns for the resource identified by the URI. The names of all System Query Options are prefixed with a "$" character.

An OData service may support some or all of the System Query Options defined. If a data service does not support a System Query Option, it must reject any requests which contain the unsupported option as defined by the request processing rules.

1.3.2 Orderby System Query Option ($orderby)

$orderby is only supported when the resource path identifies a Collection of Entities; it uses to manage the order of collection. By default, the order is ascending.

The examples below represent the most commonly supported subset of that expression syntax.

Examples

http://<hostname>/odata/v2/User?$orderby=username

- All User Entries returned in ascending order when sorted by the username Property.

http://<hostname>/odata/v2/User?$orderby=username asc

- Same as the example above.

http://<hostname>/odata/v2/User?$orderby=username,hr/username desc

- Same as the URI above except the set of User is subsequently sorted (in descending order) by the username property of the related hr Entry.

1.3.2.1 Top System Query Option ($top)

$top keyword is used to pick the topN from the identified collection.

If the data service URI contains a $top query option, but does not contain a $orderby option, then the Entries in the set needs to first be fully ordered by the data service. While no ordering semantics are mandated, to ensure repeatable results, a data service must always use the same semantics to obtain a full ordering across requests.

Examples

http://<hostname>/odata/v2/User?$top=5

- The first 5 User Entries returned where the Collection of User are sorted using a scheme determined by the OData service.

http://<hostname> /odata/v2/User?$top=5&$orderby=username desc

- The first 5 User Entries returned in descending order when sorted by the username property.

1.3.2.2 Skip System Query Option ($skip)

A data service URI with a $skip System Query Option identifies a subset of the Entries in the Collection of Entries identified by the Resource Path section of the URI. That subset is defined by seeking N Entries into the Collection and selecting only the remaining Entries (starting with Entry N+1). N is a positive integer as specified by this query option. If a value less than 0 is specified, the URI should be considered malformed.

If the data service URI contains a $skip query option, but does not contain a $orderby option, then the Entries in the Collection must first be fully ordered by the data service. While no ordering semantics are mandated, to ensure repeatable results a data service must always use the same semantics to obtain a full ordering across requests.

Examples

http://<hostname>/odata/v2/User(1)/proxy?$skip=2

- The set of proxy Entries (associated with the User Entry identified by key value 1) starting with the third User.

http://<hostname> /odata/v2/User?$skip=2&$top=2&$orderby=username

- The third and fourth User Entry from the collection of all User entities when the collection is sorted by username (ascending).

1.3.2.3 Filter System Query Option ($filter)

A URI with a $filter System Query Option identifies a subset of the Entries from the Collection of Entries. The subset is determined by selecting only the Entries that satisfy the predicate expression specified by the query option.

The expression language that is used in $filter operators supports references to properties and literals. The literal values can be strings enclosed in single quotes, numbers and boolean values (true or false) or any of the additional literal representations.

The operators supported in the expression language are shown in the following table.

Operator	**Description**	**Example**
Logical Operators		
Eq	Equal	/User?$filter=hr/username eq 'cgrant' to find User whose hr/username is equal to ‘cgrant’, hr is the navigation property /User?$filter=username eq ‘cgant’ to find User with its property equal to ‘cgrant’
Ne	Not equal	/ User?$filter=hr/username ne 'London'
Gt	Greater than	/ PicklistLabel?$filter=id gt 20000 to find PicklistLabel whose id is greater than 20000
Ge	Greater than or equal	/ PicklistLabel?$filter=id ge 10
Lt	Less than	/ PicklistLabel?$filter=id lt 20
Le	Less than or equal	/ PicklistLabel?$filter=id le 100
And	Logical and	/ PicklistLabel?$filter=id le 1005 and id gt 1000 to find PicklistLabel whose id is within range [1000,1005]
Or	Logical or	/ PicklistLabel?$filter=id le 3.5 or id gt 200

Not	Logical negation	/User?$filter=not endswith(username,'grant')
Arithmetic Operators		
Add	Addition	/PicklistLabel?$filter=id add 5 gt 10
Sub	Subtraction	/ PicklistLabel?$filter=id sub 5 gt 10
Mul	Multiplication	/ PicklistLabel?$filter=id mul 2 gt 2000
Div	Division	/ PicklistLabel?$filter=id div 2 gt 4
Mod	Modulo	/ PicklistLabel?$filter=id mod 2 eq 0
Grouping Operators		
( )	Precedence grouping	/ PicklistLabel?$filter=(id sub 5) gt 10
Customized Operators		

in	In clause	/User?$filter=userId in 'ctse1','mhuang1','flynch1'&$select=username,userId Identifies Users with whose key is equal to one of specified in in_clause
like	Like clause	/User?$filter=userId like’remy’ Identifies Users with whose userId equal to remy, equivalent to $filter=userId eq ‘remy’ /User?$filter=userId like’remy%’ Identifies Users with whose userId starts with remy, equivalent to $filter=startswith(userId, ‘remy’) /User?$filter=userId like’%remy’ Identifies Users with whose userId endswith with remy, equivalent to $filter=endswith(userId, ‘remy’) /User?$filter=userId like’%remy%’ Identifies Users with whose userId contains string remy. /User?$filter=tolower(userId) like’%remy%’ Identifies Users with whose userId contains remy in case insencisive. /User?$filter=toupper(userId) like’%remy%’ Some as previous /User?$filter=toupper(userId) like’%remy%’ or tolower(username) like ‘%remy%’ Identifies Users with whose userId like remy union Users with whose username like remy /User?$filter=toupper(userId) like’%remy%’ and tolower(username)

		like '%remy%' Identifies Users with whose userId like remy and username like remy

In addition to operators, a set of functions are also defined for use with the filter query string operator. The following table lists the available functions. Note: ISNULL or COALESCE operators are not defined. Instead, there is a null literal which can be used in comparisons.

OData_root=http://<hostname>/odata/v2

Function	Example
String Functions	
bool endswith(string p0, string p1)	OData_root/ User?$filter=endswith(username, 'Futterkiste')
bool startswith(string p0, string p1)	OData_root/ User?$filter=startswith(username, 'Alfr')
string tolower(string p0)	OData_root/ User?$filter=tolower(username) eq 'alfreds futterkiste'
string toupper(string p0)	OData_root/ User?$filter=toupper(username) eq 'ALFREDS FUTTERKISTE'
string trim(string p0)	OData_root/ User?$filter=trim(username) eq 'Alfreds Futterkiste'

1.3.2.4 Expand System Query Option ($expand)

A URI with a $expand System Query Option indicates that Entries associated with the Entry or Collection of Entries identified by the Resource Path section of the URI must be represented inline (i.e. eagerly loaded). For example, if you want to identify a User and its proxy, you could use two URIs (and execute two requests), one for /User(1) and one for /User(1)/proxy. The '$expand' option allows you to identify related Entries with a single URI such that a graph of Entries could be retrieved with a single HTTP request.

The syntax of a $expand query option is a comma-separated list of Navigation Properties. Additionally each Navigation Property can be followed by a forward slash and another Navigation Property to enable identifying a multi-level relationship.

Note: The $filter section of the normative OData specification provides an ABNF grammar for the expression language supported by this query option.

Examples

http://<hostname>/odata/v2/User?$expand=proxy

- Identifies the Collection of User as well as each of the proxy associated with each User.
- Is described by the Entity Set named "User" and the "proxy" Navigation Property on the "User" Entity Type in the service metadata document.

http://<hostname>/odata/v2/User?$expand=hr/matrixManager

- Identifies the Collection of User as well as each of the hr associated with each User. In addition, the URI also indentifies the matrixManager associated with each hr.
- Is described by the Entity Set named "User", the "hr" Navigation Property on the "User" Entity Type, and the "matrixManager" Navigation Property on the "hr" Entity Type in the service metadata document.

http://<hostname>/odata/v2/User?$expand=hr,proxy

- Identifies the set of User as well as the hr and proxy associated with each User.
- Is described by the Entity Set named "User" as well as the "hr" and "proxy" Navigation Property on the "User" Entity Type in the service metadata document.

1.3.2.5 Format System Query Option ($format)

A URI with a $format System Query Option specifies that a response to the request MUST use the media type specified by the query option. If the $format query option is present in a request URI it takes precedence over the value(s) specified in the Accept request header. Valid values for the $format query string option are listed in the following table.

$format Value	Response Media Type
atom	application/atom+xml
json	application/json

Examples

http://<hostname>/odata/v2/User?$format=atom

- Identifies all User Entries represented using the AtomPub format

```
<?xml version='1.0' encoding='utf-8'?>
<feed xmlns="http://www.w3.org/2005/Atom"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xml:base="http://<hostname>/odata/v2/">
    <title type="text">Picklist</title>
    <id>http://<hostname>/odata/v2/Picklist</id>
    <updated>2013-07-29T07:10:44Z</updated>
    <link rel="self" title="Picklist" href="Picklist" />
    <entry>
            <id>http://<hostname>/odata/v2/Picklist('CandidateStatus')</id>
            <title type="text" />
            <updated>2013-07-29T07:10:44Z</updated>
            <author>
                    <name />
            </author>
            <link rel="edit" title="Picklist" href="Picklist('CandidateStatus')" />
            <link
rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/picklistOptions"
type="application/atom+xml;type=entry" title="picklistOptions"
href="Picklist('CandidateStatus')/picklistOptions" />
            <category term="SFOData.Picklist"
scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
            <content type="application/xml">
                    <m:properties>
                            <d:picklistId>CandidateStatus</d:picklistId>
                    </m:properties>
            </content>
    </entry>
</feed>
```

http://<hostname>/odata/v2/User?$format=json

- Identifies all User Entries represented using the JSON format

```
{
    "d": {
            "results": [{
                    "__metadata": {
                            "uri": "http://<hostname>/odata/v2/Picklist('CandidateStatus')",
                            "type": "SFOData.Picklist"
                    },
                    "picklistId": "CandidateStatus",
                    "picklistOptions": {
                            "__deferred": {
                                    "uri": "Picklist('CandidateStatus')/picklistOptions"
                            }
                    }
            }]
```

```
        }
    }
```

1.3.2.6 Select System Query Option ($select)

A data service URI with a $select System Query Option identifies the same set of Entries as a URI without a $select query option; however, the value of $select specifies that a response from an OData service should return a subset of the Properties which would have been returned had the URI not included a $select query option.

Version Note: This query option is only supported in OData version 2.0 and above.

The value of a $select System Query Option is a comma-separated list of selection clauses. Each selection clause may be a Property name, Navigation Property name, or the "*" character. The following set of examples uses the data sample data model available at http://<hostname>/odata/v2/$metadata to describe the semantics for a base set of URIs using the $select system query option. From these base cases, the semantics of longer URIs are defined by composing the rules below.

Examples

http://<hostname>/odata/v2/User?$select=username,userId

- In a response from an OData service, only the username and userId Property values are returned for each User Entry.
- If the $select query option had listed a Property that identified a Complex Type, then all Properties defined on the Complex Type must be returned.

http://<hostname>/odata/v2/User?$select=username,proxy

- In a response from an OData service only the username Property value and a link to the related proxy Entry should be returned for each User.

http://<hostname>/odata/v2/User?$select=username,proxy&$expand=proxy/hr

- In a response from an OData service, only the username of the User Entries should be returned, but all the properties of the Entries identified by the proxy and hr Navigation Properties should be returned.

http://<hostname>/odata/v2/User?$select=username,proxy/username&$expand=proxy

- In a response from an OData service, the username property is included and proxy Entries with username property is included; however, rather than including the fully expanded proxy Entries, each proxy will contain a user property.

1.3.3 Custom Query Option

Custom Query Options provide an extension point for OData service-specific information to be placed in the query string portion of a URI. A Custom Query String option is defined as any name/value pair query string parameter where the name of the parameter does not begin with the "$" character. Any URI exposed by an OData service may include one or more Custom Query Options.

Examples

http://<hostname>/odata/v2/User?purge=true

- Identifies all User entities. Includes a Custom Query Option "purge" whose meaning is service specific.

1.4 Advanced Query

1.4.1 Introduction to general query

URI1 = scheme serviceRoot "/" entitySet

- Identify entities under specified entitySet. If User entitySet was given here, then return all Users in the response.

URI2 = scheme serviceRoot "/" entitySet "(" keyPredicate ")"

- Identify specific entity with given key under entitySet. If URI2 likes serviceRoot/User('1'), it identifies entity in the User entity set with the Entity Key 1

URI3 = scheme serviceRoot "/" entitySet "(" keyPredicate ")/"entityComplexProperty

- Example : URI: http://host/service.svc/User('1')/proxy
- Identifies: The value of the proxy property of the User entity identified by key value 1 in the User Entity Set.

URI4 = scheme serviceRoot "/" entitySet "(" keyPredicate ")/" entityComplexProperty "/" entityProperty

- Example: http://host/service.svc/User('1')/proxy/username
- Identifies: The value of the username property of the proxy ComplexType property of the User entity identified by key value 1 in the User Entity Set.

URI4-2 = scheme serviceRoot "/" entitySet "(" keyPredicate ")/" entityComplexProperty "/" entityProperty/$value

- Same as URI4, but it only returns raw string. And last property prior to $value should be in primitive type.

- Example : http://host/service.svc/User('1')/proxy/username/$value
- Identifies: The raw value of the username property of the proxy ComplexType property of the User entity identified by key value 1 in the User Entity Set. Return value should looks like "rocky", should have no surrounding envelop.

URI5 = scheme serviceRoot "/" entitySet "(" keyPredicate ")/" entityProperty

- Example: URI: http://host/service.svc/User('ALFKI')/username
- Identifies: The name of the User entity in the User EntitySet identified by key 'ALFKI'.

URI5 = scheme serviceRoot "/" entitySet "(" keyPredicate ")/" entityProperty/$value

- Example: URI: http://host/service.svc/User('ALFKI')/username/$value
- Identifies: Same as proceeding, but identifies the value of the property free of any metadata or surrounding markup.

URI6 = scheme serviceRoot "/" entitySet "(" keyPredicate ")/" entityNavProperty

- Example: URI: serviceRoot/PicklistLabel(optionId=498L,locale='en_US')/picklistOption
- Identifies: The set of picklistOption Entity Type instances (or instances of a sub type of picklistOption) associated with the PicklistLabel identified by the key (optionId=498L,locale='en_US')through the picklistOption Navigation Property. If the key is singular key, just specify like User('userId');

URI7 = scheme serviceRoot "/" entitySet "(" keyPredicate ")/$links/" entityNavProperty

- Example: http://host/odata/v2/PicklistLabel(optionId=498L,locale='en_US')/$links/picklistOption
- Identifies: The collection of all Links between the entity in the PicklistLabel Entity Set identified by key (optionId=498L,locale='en_US') and the picklistOption entities associated with that PicklistLabel via the picklistOption navigation property.
- URI: http://host/odata/v2/User('1')/$links/proxy
- Identifies: The Link between the User entity with key value 1 in the User Entity Set and proxy entity associated with that User via the proxy navigation property.

URI8 = scheme serviceRoot "/$metadata"

- Comment: URI: http://host/odata/v2/$metadata
- Identifies: The EDMX (metadata) document for the data service

Here is a table to summarize the general URI works with which system query options

	$orderby	$top	$skip	$filter	$expand	$format	$select
URI1	Yes	Yes	Yes	Yes	Yes	Yes	Yes
URI2	Yes				Yes	Yes	Yes
URI3						Yes	Yes
URI4						Yes	
URI5						Yes	
URI6	Yes				Yes	Yes	Yes
URI7						Yes	
URI8							

1.4.2 Comprehensive Examples

In this section, it demonstrates how to query with combination of system query options

1.4.2.1 Query with nested-navigation properties with $select

Uri_a. http://<hostname>/odata/v2/User?$select=userId,username,hr&$format=json&$top=1

Uri_b.
http://<hostname>/odata/v2/User?$select=userId,username,hr/manager/proxy&$format=json&$top=1

- Identifies all entities of User. In response there wouldn't show all properties of each entity, because $select is given here. In each entity, only three selected properties would show in the result. Selected property could be simple type or navigation, or even nested-navigation property. For simple type properties will show in result directly, for navigation property, it would show a value as a deferred link. Whatever depth the navigation property navigates, it always shows a deferred link for the navigation property. Therefore, uri_a and uri_b get the same result because they have selected exactly the same 2 simple properties and one navigation property. $top here to limit the size of response.
- Example response:

```
{
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('mhuang1')", "type" : "SFOData.User"
}, "userId" : "mhuang1", "username" : "mhuang", "manager" : {
    "__deferred" : {
    "uri" : "http://localhost:8080/odata/v2/User('mhuang1')/manager"
    }
}
}
```

1.4.2.2 Query with $select and $expand

http://<hostname>/odata/v2/User?$select=userId,manager/hr/manager&$format=json&$expand=manager/hr

- Identifies all entities of User entitySet. The response is quite similar to the previous link, but there is slight difference. If the navigation property gets expanded, then it shows as a flat object, otherwise, it shows as a deferred link. Selected nested-navigation property manager/hr/manager gets to be expanded, then it will show an object manager, and manger gets a nested object hr, because last manager of selected nested-navigation property doesn't get expanded, then the hr has a deferred link named manager which refers to the real object. For the unselected properties of expanded object will be trimmed.
- Example response:

```
{
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('rocky_upsert_JAM_200')", "type" : "SFOData.User"
}, "userId" : "rocky_upsert_JAM_200", "manager" : {
    "__metadata" : {
    "uri" : "http://localhost:8080/odata/v2/User('rocky_upsert_JAM_201')", "type" : "SFOData.User"
    },
    "hr" : {
                    "__metadata" : {
                    "uri" :
"http://localhost:8080/odata/v2/User('rocky_upsert_b11_112_inline_manager11')", "type" :
"SFOData.User"
                    },
                    "manager" : {
                            "__deferred" : {
                                    "uri" :
"http://localhost:8080/odata/v2/User('rocky_upsert_b11_112_inline_manager11')/manager"
                                    }
                            }
            }
    }
}
}
```

1.4.2.3 Query links with $select and $expand

http://<hostname>/odata/v2/User('rocky_upsert_JAM_200')/proxy?$select=userId,manager/hr/manager&$format=json&$expand=manager/hr

- It's hard to expect what would return in response when it comes to you in the first glance. Here is a simple principle, uri part goes first, then $select goes, last $expand goes. Last uri part tells it will return set entities of User/proxy. Then $select tells unselected could should be trimmed, last $expand will expand selected navigation property. Unselected column won't be expanded even $expand token is provided.

- Example response: it shows that, User('rocky_upsert_JAM_200') has four proxy, select properties are userId, manager/hr/manager, because manager is null, so $expand wont' be executed.

```
[
{
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('rocky_upsert_JAM_197')", "type" : "SFOData.User"
}, "userId" : "rocky_upsert_JAM_197", "manager" : null
},

{
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('rocky_upsert_JAM_198')", "type" : "SFOData.User"
}, "userId" : "rocky_upsert_JAM_198", "manager" : null
},

{
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('rocky_upsert_b11_112_inline_proxy')", "type" :
"SFOData.User"
}, "userId" : "rocky_upsert_b11_112_inline_proxy", "manager" : null
},

{
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('rocky_upsert_b11_112_inline_proxy_1')", "type" :
"SFOData.User"
}, "userId" : "rocky_upsert_b11_112_inline_proxy_1", "manager" : null
}
]
```

1.4.2.4 Query with $select and $expand and $filter

http://<hostname>/odata/v2/User?$select=userId,manager/hr/manager&$format=json&$expand=manager/hr&$filter=proxy/userId eq 'rocky_upsert_b11_112_inline_proxy_1'

- The request identifies a set of User with whose proxy goes with userId equal to 'rocky_upsert_b11_112_inline_proxy_1' . After proper entities are retrieved, then remove $unselect column, do $expand if the selected columns is navigation property.

- Example response:

```
[
{
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('rocky_upsert_JAM_200')", "type" : "SFOData.User"
}, "userId" : "rocky_upsert_JAM_200", "manager" : {
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('rocky_upsert_JAM_201')", "type" : "SFOData.User"
}, , "hr" : {
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('rocky_upsert_b11_112_inline_manager11')", "type" :
"SFOData.User"
}, , "manager" : {
"__deferred" : {
"uri" : "http://localhost:8080/odata/v2/User('rocky_upsert_b11_112_inline_manager11')/manager"
}
}
}
}
},

{
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('rocky_upsert_JAM_201')", "type" : "SFOData.User"
}, "userId" : "rocky_upsert_JAM_201", "manager" : {
"__metadata" : {
"uri" : "http://localhost:8080/odata/v2/User('rocky_upsert_JAM_198')", "type" : "SFOData.User"
}, "hr" : null
}
}
]
```

1.4.2.5 Query with $top and $skip

If $top and $top token goes together in the request URI, it will do a pagination query, in response it will get subset of whole result, from $skip to $top. If $top=50, $skip=20, it means to get a subset from number 21 to number 70 of the whole result set. Next chapter has more details about pagination.

1.5 Pagination

OData API provides the simple pagination for the Query results - it limits the maximum result size to 800 in one query response. It is used to protect the server from crash by some heavy query and meanwhile make sure the client side can get a sub set of the results instead of the whole result set.

1.5.1 How the pagination works in OData

We followed the OData standard to provide a '__next' link in your query response if you still have more results in the DB, below are the sample request and response for User Query:

➢ Request:

GET http://<hostname>/odata/v2/User?$format=json

➢ Response:

```
{
"d" : {
"results" :
 ................
 "__next" : "http://
<hostname>/odata/v2/User?$format=json&$skiptoken=eyJzdGFydFJvdyI6NSwiZW5kUm93IjoxMH0%3D"
}
}
```

$skiptoken will contain the information to indicate the startRow and endRow of your entity records in your next query. Directly copy the URL and access it, you will get the next 800 hundreds records if you have.

➢ $skip and $top

Both $skip and $top goes together in the URI will be considered a pagination query. Skip will indicate the start row number of your entity records, if there is no skip then the start row number will be 0. $top will limit the result size in your query. If $top value is less than 800 then in the query response it will only return the number of the records which equals to the $top value.

➢ Exceptional Cases

Note that OData supports RBP row and field level permission check, so it could happen that in one query request you don't have permission to view any row/field but it still returns the next link to indicate the next 800 records.
Now the page size is fixed to 800 in b1308 and can't be change. In b1311 we will implement an advanced pagination strategy.

Copyright