CLOUD DATA OBJECT CATALOG SPECIFICATION

Progress is committed to delivering market-leading technology innovations that empower our partners and customers to dramatically improve the development, deployment, integration and management of their business applications. The Cloud Data Object brings data management features normally found in Systems of Records to client applications in the cloud, specifically web and mobile apps.

Server samples in this specification use the OpenEdge ABL but any language that can be exposed using REST can be used on the backend server.

Document History:

VERSION	DATE	CHANGE DESCRIPTION
1.0	05/15/2015	Initial version
1.1	12/15/2015	corrected examples and some typos

Contents

Introduction

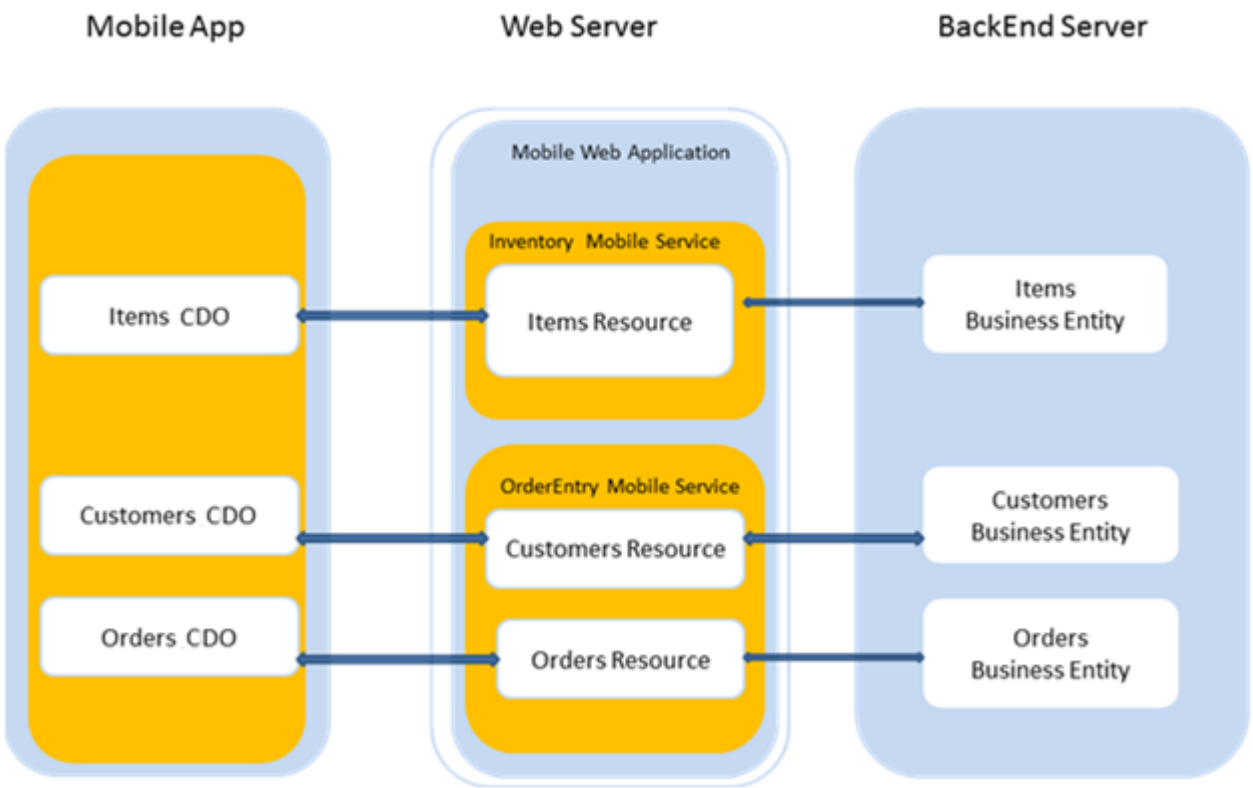
Mobile applications often communicate with a server for data and business logic. This specification introduces the *Cloud Data Object* (CDO), a client-side object that manages transactional data updates and access to server-side business logic.

A *Cloud Data Service* (CDS) defines the API for one or more server-side resources. Each resource provides access to a logical schema and its related business logic, which for the purposes of this document will be referred to as a *Business Entity*. The supported schemas are a single table or a dataset which contains one or more tables where the tables may be related. A Business Entity is implemented using a standard set of built-in operations to read and modify data on the backend. A Business Entity can also provide additional (customized) methods, so the resource will also provide access to these non-standard methods.

A CDO provides client access to the data and operations of a single resource. Client code can call methods on a CDO to execute the mapped server-side operations on the backend. The data for these operations is serialized between the Client and the Server.

For information on the *CDO API,* please refer to the document Cloud Data Object API Specification.

The CDO needs a means to acquire meta-data about the resource i.e. the resource's schema and its methods (or operations). This information is provided in the *CDO Catalog file.*



The figure above provides an overview of how a CDO accesses a resource which works the same for a given CDO regardless of the type of client or type of backend.

Mapping CDO Operations to Server-side Objects

Transactional Update Support

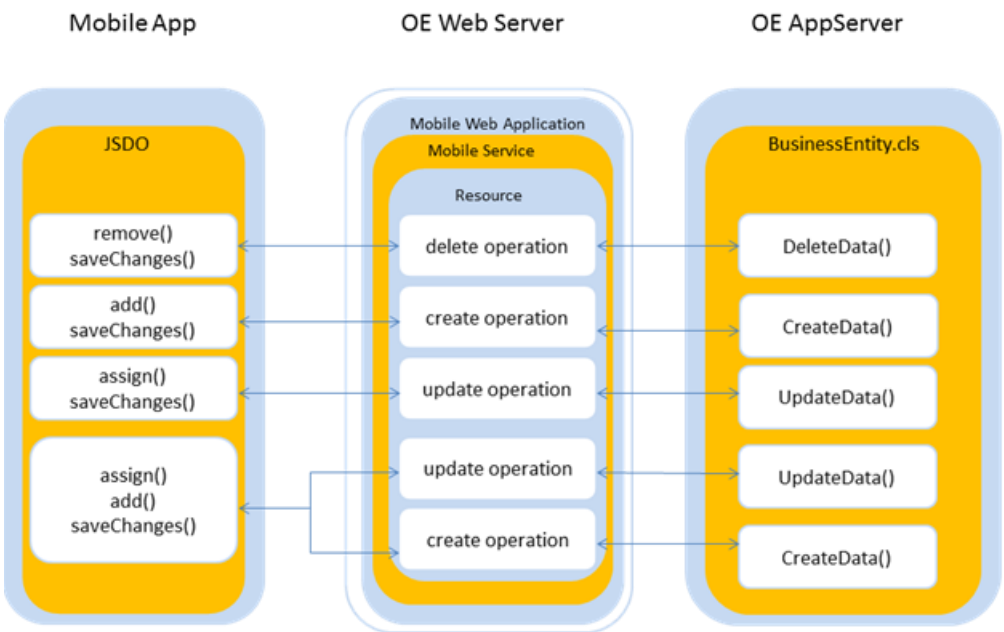
As part of the built-in CRUD operations, a Business Entity can have individual Create, Update and Delete methods or a single "*Submit*" method. A "*Submit*" method allows the developer to design their mobile app such that changes made on client can be sent back to the backend server in a single transaction.

Single Record Update

It is important to understand how the CDO sends CUD operations (create, update, delete) to the server. When the CUD methods are called (add( ), remove( ) and assign()) on the CDO, the changes are made to the CDO memory. No communication with the server is performed at this time.

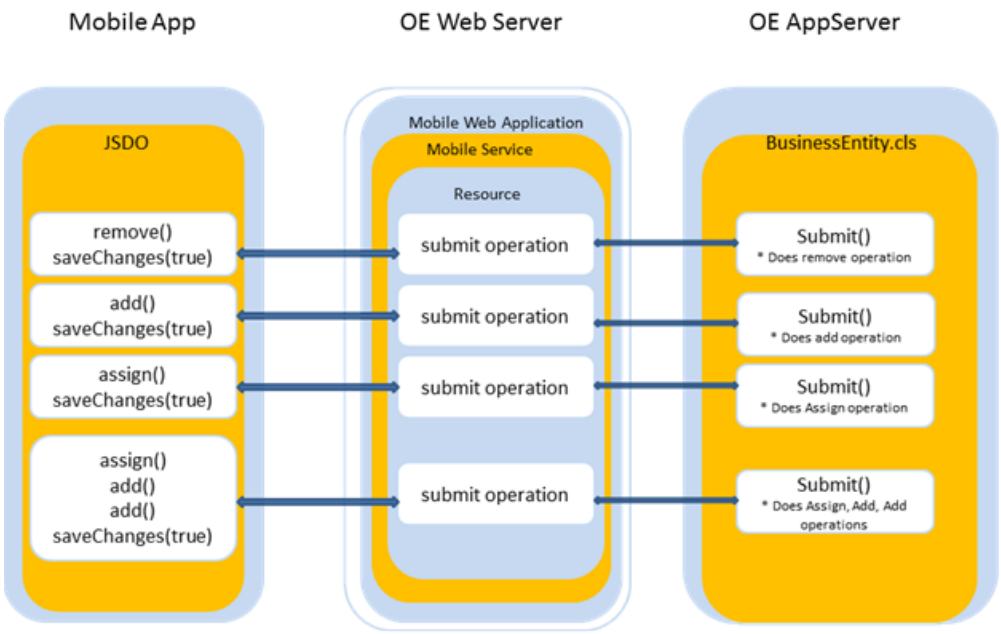
To send the changes to the backend server, the CDO saveChanges() or saveChanges(false) method should be called. For each change, the CDO sends a single row to its corresponding Business Entity method. In the diagram below you can see that the first three updates call saveChanges( ) immediately after calling remove( ), assign( ) and add( ). Each record is sent to the server individually. Even if more than one update is performed before calling saveChanges), each update is sent separately.

In the last example below, assign( ) and add( ) are called followed by a single call to saveChanges(false). SaveChanges( ) will send a separate request for each record to the server. This results in two separate transactions, the first to UpdateData( ) and the next to CreateData( ) on the Business Entity.



Multiple Record Update

In order to group records into a single operation, you can call the saveChanges( ) method with "true". This will send all updates since the last communication with the server to the Submit( ) method on the Business Entity in a single request. The next diagram depicts a client designed with the saveChanges(true) method. The first three updates call saveChanges(true) immediately after calling remove( ), assign( ) and add( ) just like above. Since there is only a single record of data since the last call to the server, each record is still sent to the server individually but in all three cases the Submit( ) method is called on the Business Entity. In the last example below assign( ), add( ), add() are called before saveChanges(true) is called. This results in saveChanges(true) sending all three record updates in a single call to Submit() so it can performs all three changes to the backend data as a single transaction.



Note: When saveChanges() is specified with no parameter, useSubmit param is false.

Built-In CRUD Operations

As mentioned above, a Business Entity can be implemented with a standard set of methods to read and modify its data. These methods are referred to as CRUD operations (**C**reate, **R**ead, **U**pdate, **D**elete). Each of these methods has a prescribed signature. Since the interface for each CRUD operation is prescribed, the CDO can provide a set of built-in methods that correspond to the Business Entity CRUD methods. For the CRUD operations to work successfully together, the data (and its schema) are shared between the operations. No validation of this requirement occurs on the client.

Read Operation

The read method of a Business Entity is used to return records that are loaded into the CDO memory.

Business Entity required signature:

- An input parameter of type string. Parameter must be named *filter*
- An output parameter corresponding to a Table or DataSet (specified in schema)

Example OpenEdge ABL method:

```
METHOD PUBLIC VOID ReadOrders(INPUT filter AS CHARACTER,
                              OUTPUT DATASET dsOrder):
END METHOD.
```

CDO Catalog snippet:

```
"operations": [
    {
        "path": "?filter={filter}",
        "useBeforeImage": true,
        "type": "read",
        "verb": "get",
        "params": [
            {
                "name": "filter",
                "type": "PATH",
                "xType": "CHARACTER"
            },
            {
                "name": "dsOrder",
                "type": "RESPONSE_BODY",
                "xType": "DATASET"
            }
         ]

    },…
]
```

Create Operation

The create method of a Business Entity is called when a new record is sent by the CDO to the server. This is executed when CDO's saveChanges(false) is called.

Business Entity required signature:

- Input-output param corresponding to a Table or DataSet (specified in schema in catalog)

Example OpenEdge ABL method:

```
METHOD PUBLIC VOID CreateOrder(INPUT-OUTPUT DATASET dsOrder):
END METHOD.
```

CDO built-in method:

- `CDO.saveChanges(false)` useSubmit param is set to false

CDO Catalog snippet:

```
"operations": [
    {
        "path": "",
        "useBeforeImage": true,
        "type": "create",
        "verb": "post",
        "params": [{
            "name": "dsOrder",
            "type": "REQUEST_BODY,RESPONSE_BODY",
            "xType": "DATASET"
        }]
    },…
]
```

Update Operation

The update method of a Business Entity is called when an existing record update is sent by the CDO to the server. This is executed when CDO's saveChanges(false) is called.

Business Entity required signature:

- Input-output param corresponding to a Table or DataSet (specified in schema in catalog)

Example OpenEdge ABL method:

```
METHOD PUBLIC VOID UpdateOrder(INPUT-OUTPUT DATASET dsOrder):
END METHOD.
```

CDO built-in method:

- `CDO.saveChanges(false)` useSubmit param is set to false

CDO Catalog snippet:

```
"operations": [
    {
        "path": "",
        "useBeforeImage": true,
        "type": "update",
        "verb": "put",
        "params": [{
            "name": "dsOrder",
```

```
            "type": "REQUEST_BODY,RESPONSE_BODY",
            "xType": "DATASET"
        }]
    },…
]
```

Delete Operation

The delete method of a Business Entity is called when an existing record delete is sent by the CDO to the server. This is executed when CDO's saveChanges(false) is called.

Business Entity required signature:

- Input-output param corresponding to a Table or DataSet (specified in schema in catalog)

Example OpenEdge ABL method:

```
METHOD PUBLIC VOID DeleteOrder(INPUT-OUTPUT DATASET dsOrder):
END METHOD.
```

CDO built-in method:

- `CDO.saveChanges(false)` useSubmit param is set to false

CDO Catalog snippet:

```
"operations": [
    {
        "path": "",
        "useBeforeImage": true,
        "type": "delete",
        "verb": "delete",
        "params": [{
            "name": "dsOrder",
            "type": "REQUEST_BODY,RESPONSE_BODY",
            "xType": "DATASET"

        }]
    },…
]
```

Submit Operation

A Business Entity can also perform updates as part of a transaction by defining a Submit method. This extension is necessary over normal REST APIs since enterprise businesses often need to group records updates rather than update a single record at a time.
For transaction support, the server will often start a transaction, process all record updates and then submit the transaction. The main difference here from the CRUD operations is that the record update sends multiple records in a single request to the server.

Submit Operation

The Submit method of a Business Entity is called when a group of new, changed and deleted records are sent by the CDO to the server. This is executed when CDO's saveChanges(true) is called.

Business Entity required signature:

- Input-output param corresponding to a Table or DataSet (specified in schema in catalog)

Example OpenEdge ABL method:

```
METHOD PUBLIC VOID SubmitOrderChanges(INPUT-OUTPUT DATASET dsOrder):
END METHOD.
```

CDO built-in method:

- `CDO.saveChanges(true)` useSubmit param is set to true

CDO Catalog snippet:

```
"operations": [
    {
        "name": "SubmitOrderChanges",
        "path": "/SubmitOrderChanges",
        "useBeforeImage": true,
        "type": "submit",
        "verb": "put",
        "params": [{
            "name": "dsOrder",
            "type": "REQUEST_BODY,RESPONSE_BODY",
            "xType": "DATASET"
        }]
    },
]
```

Invoke Operations

A Business Entity can also implement customized invocation methods. This extension is necessary over normal REST APIs since enterprise businesses often have business logic that is not part of a CRUD operation such as checking a customer's credit limit or complicated pricing details for an order.

The signature of an invocation method is unrestricted and can have both input and output parameters. These methods can work with the Business Entity schema and data, or work on other data, there is no requirement.

Example OpenEdge ABL method:

```
METHOD PUBLIC VOID GetCreditLimit(INPUT custNum AS INT, OUTPUT TABLE eTable):
END METHOD.
```

CDO method:

- `CDO.GetCreditLimit( { custNum: 10 } );`

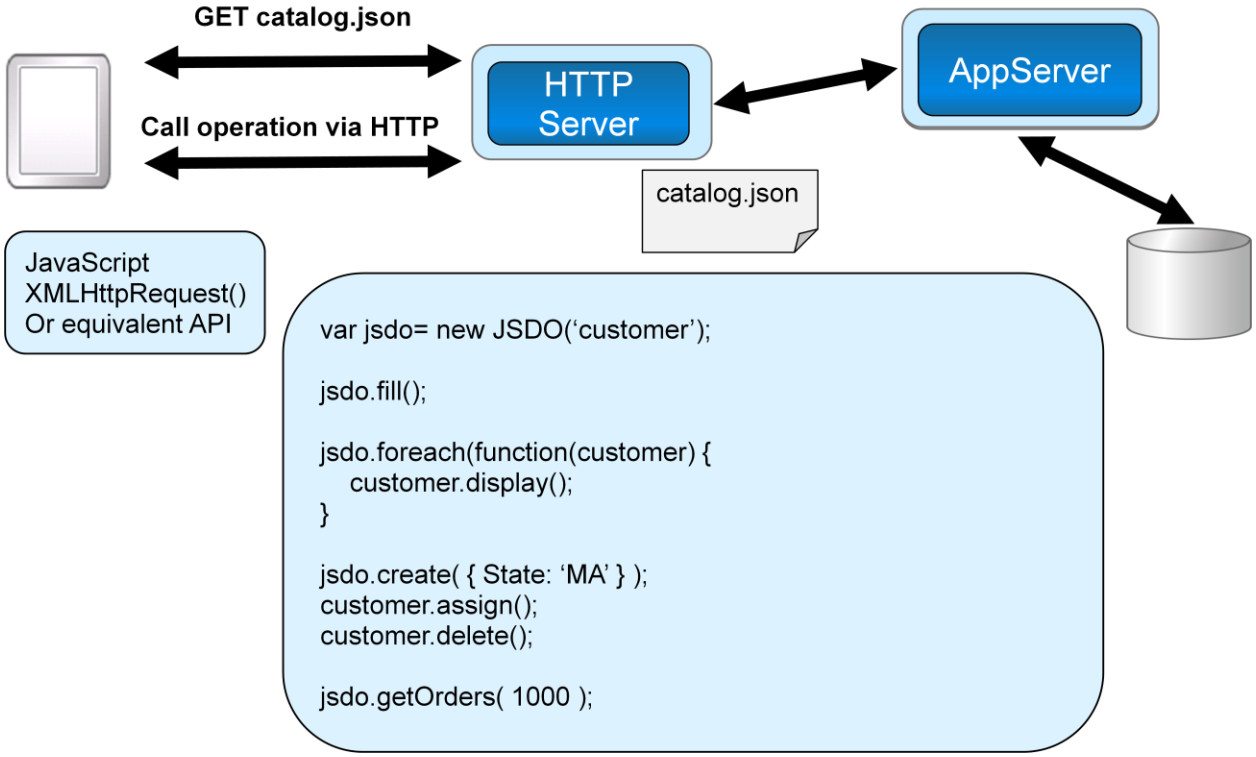
CDO Catalog snippet:

```
"operations": [
    {
        "name": "GetCreditLimit",
        "path": "/GetCreditLimit",
        "useBeforeImage": false,
        "type": "invoke",
        "verb": "put",
        "params": [
            {
                "name": "custNum",
                "type": "REQUEST_BODY"
            },
            {
                "name": "eTable",
                "type": "RESPONSE_BODY",
                "xtype": "TABLE"
            },
        ]
    },
]
```

The CDO Catalog File

The CDO Catalog includes an entry for each built-in method and each invocation (or invoke) method for the resource. It also includes the schema for the resource. The CDO uses the schema information for its built-in CRUD methods. For each method, the catalog provides parameter mapping information. This information (metadata) from the catalog is used to construct the HTTP request for the operation calls made to the REST adapter.

The catalog also allows the CDO to be hypertext driven, so that URIs are not hardcoded in the client application.



The above diagram depicts a backend using the OpenEdge AppServer as an example.

The format of the catalog file is the following:

```
{
    "version": <Format version>,
    "lastModified": “<Generation Date>”,
    "services": [
        {  "name": "<Service Name>",
           "address": "/rest/<Service Name>",

           "tenantId": "<tenantId>",
           "appId": "<appId>"
           "settings": {
               "useRequest": <true|false>,
               "sendOnlyChanges": <true|false>,
               "unwrapped": <true|false>,
               "useXClientProps": <true|false>
            },
           "resources": [
                {
                    "name": “<Resource Name>”,
                    "path": "<Relative URI for the Resource>",

                    "idProperty": "<field name>",
                    "displayName": "<displayName>",
                    "schema": { Schema Formats },
                    "dataDefinitions": { Data Definition },
                    "relations": [ Relations Property ],
                    "operations": [ Operations Property ]
                }
            ]
        }
    ]
}
```

Catalog Property Descriptions

<format version>	Catalog format version number. Initial version is 1.0. Current version is 1.3.
<Generation Date>	Date when file is generated

The services Property Descriptions

<Service Name>	Name of the Service.
useRequest	Specifies whether a request object should be used for invoke operations. Default value is false. OE backends require this to be set to true.

sendOnlyChanges	Optional.If true, tells CDO to only send the fields that change back to the server. If false, CDO will send the whole record. Default is false.
unwrapped	Optional. Indicates whether the server expects an enclosing object of the same name when the CDO sends parameters.
useXClientProps	Optional. Indicates whether or not the backend expects a clientProps header with context for every operation.
tenantId	Optional. Represents the customer id.
appId	Optional. Represents the application id.

.

The resources Properties

<Resource Name>	Name of resource containing business logic
<Relative URI for the Resource>	Unique resource URI, relative to its service address
idProperty	Optional. Indicates to the CDO the name of the field in the schema that should be used to identify a record. This has been added because some types of services have a specific property (field) to identify records. If specified, it is sent to the server as part of a Create, Update, or Delete operation. It is also used to match records when merging data from an Invoke operation into the CDO memory.
displayName	Optional. A client tool can use this for display purposes in places where a name for a resource needs to be shown. If not specified, client tools can use the resource's "name" property.

Schema Formats

The Schema property provides the data definition to be used by the built-in CRUD operations. The valid schema formats are listed below.

Table Schema

```
"schema": {
    "type": "object",
    "additionalProperties":false,
    "properties": {
    "<TableName>": {\
        "primaryKey": ["<Field1>"],
        "recordName": "<FieldName>",
        "type": "array",
            "items": {
                "additionalProperties":false,
                "properties": {
                    "_id": { "type": "string"},
                    "_errorString": { "type": "string"},
                    "<Field1>": {
                        "default": <Default Value>,
                        "type": "<JSON Schema DataType>",
                        "title": "<Label>",
                        "format": "<FormatType>",
                        "ablType": "<ABLType>",
                        "required": true/false,
                       "readOnly": true/false
                    },
                    "<Field2>": {
                        "type": "<JSON Schema DataType>",
                        "title": "<Label>",
                        "ablType": "<ABLType>",
                       "readOnly": true/false
                    },
                    "<Field3>": {
                        "type": "string",
                        "title": "MyBlob",
                        "ablType": "BLOB",
                        "default": null,
                        "contentEncoding": "base64",
                        "required": true/false
                    },
```

```
                    "<ArrayField>": {
                        "type": "array",
                        "title": "<Label>",
                        "maxItems": <ExtentValue>,
                        "ablType": "<ABLType>",
                        "items": {"type": "<JSON Schema DataType>"},
                          "default": <Default Value>,
                    }
                }
            }
        }
    }
}
```

DataSet Schema/Non-Nested

For a DataSet where *nested* is not specified, a table property should be specified for each table in the DataSet

```
"schema": {
    "type": "object",
    "additionalProperties": false,
    "properties": {
        "<DataSetName>": {
            "type": "object",
            "additionalProperties": false,
            "properties": {
                "<TableName1>": {
                    "recordName": "<FieldName>",
                    "type": "array",
                    "items": {
                        "additionalProperties": false,
                        "properties": {
                            "_id": {"type": "string"},
                            "_errorString": {"type": "string"},
                            "<Field1>": {
                                "default": < Default Value > ,
                                "type": "<JSON Schema DataType>",
                                "title": "<Label>",
                                "format": "<FormatType>",
                                "ablType": "<ABLType>",
                                "required": true/false,
                                "readOnly": true/false
                            },
                            "<Field2>": {
                                "type": "<JSON Schema DataType>",
                                "title": "<Label>",
```

```
                    "format": "<FormatType>",
                    "ablType": "<ABLType
                },
                "<Field3>": {
                    "type": "string",
                    "title": "MyBlob",
                    "ablType": "BLOB",
                    "default": null,
                    "contentEncoding": "base64"
                },
                "<ArrayField>": {
                    "default": < Default Value > ,
                    "type": "array",
                    "title": "<Label>",
                    "maxItems": < ExtentValue > ,
                    "ablType": "<ABLType>"
                    "items": {"type": "<JSON Schema DataType>"}
                }
            }
        }
    },

    "<TableName2>": {
        "type": "array",
        "items": {
            "additionalProperties": false,
            "properties": {
                "_id": {
                    "type": "string"
                },
                "_errorString": {
                    "type": "string"
                },
                "<Field1>": {
                    "default": <Default Value>,
                    "type": "<JSON Schema DataType>",
                    "title": "<Label>",
                    "ablType": "<ABLType>",
                    "required": true/false,
                    "readOnly": true/false
                },
                "<Field2>": {
                    "type": "<JSON Schema DataType>",
                    "title": "<Label>",
                    "format": "<FormatType>",
                    "ablType": "<ABLType>"
                },
                "<Field3>": {
```

```
                        "type": "string",
                        "title": "MyBlob",
                        "ablType": "BLOB",
                        "default": null,
                        "contentEncoding": "base64"
                    },
                    "<ArrayField>": {
                        "default": <Default Value>,
                        "type": "array",
                        "title": "<Label>",
                        "readOnly": true/false,
                        "maxItems": <ExtentValue>,
                        "ablType": "<ABLType>"
                        "items": {
                            "type": "<JSON Schema DataType>"
                        }
                    }
                }
            }
        }
    }
}
```

DataSet Schema/Nested

The following schema format should be specified in the catalog if the data exchanged between the client (CDO) and backend service (resource) is to be passed in nested format. As nested, the JSON data is formatted such that the child rows are nested within their parent rows, based upon a relationship between 2 tables in a DataSet.

Note: The following format specifies a DataSet with 2 tables with a nested data-relation. A DataSet can have multiple (more than 2) tables and multiple data-relations. The nested option is specified on the data-relation, so a DataSet's schema format can contain both nested and non-nested formats. Examples of this type of DataSet schema can be found in the example section of this document.

For a DataSet where NESTED is specified, a table property should be specified for each table in the DataSet (same as for non-nested).

Additionally, a property referencing the child table needs to be added to the parent table's property list (in bold below).

For the Child table that is involved in the Data-RELATION as **NESTED**, the table's type should be defined as an *object* (instead of an *array*). This is because the child table is referenced in its parent table as an *array* already.

```
"schema": {
    "type": "object",
    "additionalProperties": false,
    "properties": {
        "<DataSetName>": {
            "type": "object",
            "additionalProperties": false,
            "properties": {}
            "<ParentTableName>": {
                "type": "array",
                "primaryKey": ["<Field1>"],
                "foreignKeys": [{
                    "fields": ["<Field1>,FieldN"],
                    "parent": {
                        "name": "<parentName>",
                        "fields": ["<ParentField1,ParentFieldN>"]
                    }
                }],
                "indexes": "<indexes object>",
                "items": {
                    "additionalProperties": false,
                    "properties": {
                        "_id": {
                            "type": "string"
                        },
                        "_errorString": {
                            "type": "string"
                        },
                        "<Field1>": {
                            "default": < Default Value > ,
                            "type": "<JSON Schema DataType>",
                            "title": "<Label>",
                            "ablType": "<ABLType>",
                            "required": true/false
                        },
                        "<Field2>": {
                            "type": "<JSON Schema DataType>",
                            "title": "<Label>",
                            "format": "<FormatType>",
                            "ablType": "<ABLType>",
                            "required": true/false
                        },
                        "<Field3>": {
                            "type": "string",
                            "title": "MyBlob",
                            "ablType": "BLOB",
                            "default": null,
```

```
                            "contentEncoding": "base64"                         },
                        "<ArrayField>": {
                            "default": < Default Value > ,
                            "type": "array",
                            "title": "<Label>",
                            "maxItems": < ExtentValue > ,
                            "ablType": "<ABLType>"
                            "items": {
                                "type": "<JSON Schema DataType>"
                            }
                        }
                        "<ChildTableName>": {
                            "type": "array",
                            "items": {
                                "$ref":
"#/properties/<DataSetName>/properties/<ChildTableName>"
                            }
                        }
                    }
                }
            },
            "<ChildTableName>": {
                "type": "object",
                "additionalProperties": false,
                "properties": {
                    "_id": {"type": "string"},
                    "_errorString": {"type": "string"},

                    "<Field1>": {
                        "default": < Default Value > ,
                        "type": "<JSON Schema DataType>",
                        "title": "<Label>",
                        "format": "<FormatType>",
                        "ablType": "<ABLType>"
                    },
                    "<Field2>": {
                        "type": "<JSON Schema DataType>",
                        "title": "<Label>",
                        "ablType": "<ABLType>"
                    },
                    "<Field3>": {
                        "type": "string",
                        "title": "MyBlob",
                        "ablType": "BLOB",
                        "default": null,
                        "contentEncoding": "base64"
                    },
                    "<Field4>": {
```

```
                    "type": "number",
                    "title": "Balance",
                    "ablType": "DECIMAL",
                    "default": 0,
                    "width": 17,
                    "scale": 2
                },
                "<ArrayField>": {
                    "default": < Default Value > ,
                    "format": "<FormatType>",
                    "type": "array",
                    "title": "<Label>",
                    "maxItems": <ExtentValue> ,
                    "ablType": "<ABLType>"
                    "items": {"type": "<JSON Schema DataType>"}
                }
            }
        }
    }
}
```

PROPERTY	INFO
<DataSetName>	Required. DataSet Name For OE: If the ABL DataSet definition includes the SERIALIZE-NAME option, then use that as the name
<ParentTableName> <ChildTableName> <TableName>	Required. Table Name For OE: If the ABL temp-table definition includes the SERIALIZE-NAME option, then use that as the name
<Field1>....<FieldN> <ArrayField> <ParentField1>..<ParentFieldN>	Required. Field Name One or more fields can be specified for each table (comma separated) A table can have 0 or more array fields, but has to have at least 1 field, whether it's an array or scalar field For OE: If the ABL field definition includes the SERIALIZE-HIDDEN option, then don't include that field in the catalog definition
_id	Required. An internal field for the CDO that is provided to find a given record in its memory. .
_errorString	Optional. Used by the CDO to do automatic data mapping for any error string passed back from backend with before-imaging data
indexes	Optional. Specifies list of table indexes. More on indexes below in separate section.
primaryKey	Optional. Table level property that specifies the primary key of the entity (table reference). An array of strings where each value is the

	name of a field (column) in the entity.
foreignKeys	Optional. Table level property that specifies the relationships of the entity (table reference) to other entities. The other entities do not need to be defined in the same resource (for a DataSet) nor in the same catalog. More on foreign keys below.
recordName	Optional. Table level property that specifies the field to be used as the Record Name field for the backend server

Field properties:

PROPERTY	INFO
ablType	Required for OE backends. See the OpenEdge ABL datatype mapping chart (below) for details.
contentEncoding	Optional. Value would be: “base64”. Some fields (Ex. The OpenEdge BLOB, RAW, ROWID) require this property set as “base64”. See the OpenEdge ABL datatype mapping chart (below) for details.
default	Optional. Specifies default value. For OE: If the ABL field definition includes the INIT option, use that. Ex ABL: DEFINE VAR I AS INT INIT 5 Specify: “default”: 5 Ex ABL: DEFINE VAR name AS CHAR INIT “Foo”. Specify: “default”: “Foo” If INIT option is included as UNKNOWN (?), specify: “default”:null If the “INIT” option is not included, use the default value for the specified type. (See DataType Mapping Chart below.) *For Dates, DateTime,and DateTime-TZ datatypes*, see section below
format	Optional. Some fields (ex. The OE Date, DateTime and DateTimeTZ require the “format” property. See the OpenEdge ABL datatype mapping chart below for details.
maxItems	Optional for Field with type “array”. <*ExtentValue*> is an integer value specifying number of array elements.
title	Optional. Field label

	For OE: If the ABL field definition includes the LABEL option, use that first. Else if just the COLUMN-LABEL is specified, use that. If neither is specified, use the field name.
type	Optional. The <JSON Schema Data Type> can be either: “string” “integer” “number” “boolean” “array”
required	Optional. If specified, indicates whether the field can have a null value (unknown value in ABL or undefined in JSON).
readOnly	Optional. Indicates whether specified field is readOnly. Default is false.
Scale	Optional. The scale property corresponds to the numbers of digits to the right of the decimal point for decimal fields.
width	Optional. This attribute corresponds to the maximum number of digits for numeric fields (created as NUMERIC or DECIMAL).

Foreign Keys

The foreignKeys property is an array of objects where each object has a fields and parent property. The "fields" property is an array of strings where each value is the name of a field in the entity. The parent property is an object with a name and fields properties. The name corresponds to the name of the entity that the foreign key references. The "fields" property at the parent level is an array of strings where each value is the name of a field in the entity.

Note: An alternative to the foreignKey property would be to use the relations property used for a DataSet definition. However, the reason for using a new property is to avoid potential parsing issues and disambiguate the case where foreign key support is designed vs having a relationship in a DataSet.

Example of "foreignKeys" property in "eOrder":

```
"foreignKeys": [ {
    "fields": [ "CustNum" ],
    "parent": {
        "name": "eCustomer",
        "fields": [ "CustNum" ]
    }
} ]
```

indexes Property

The Optional "indexes" property specifies a list of table indexes. CDO could use indexes to optimize the query. Index has one or many fields to be indexed.

```
"schema": {
    "type": "object",
    "additionalProperties": false,
    "properties": {
        "dsCustomer": {
            "type": "object",
            "additionalProperties": false,
            "properties": {
                "eCustomer": {
                    "type": "array",
                    "primaryKey": ["CustNum"],
                    "recordName": "Name",
                    "indexes": {
                        "Comments": {
                            "wordIndex": true,
                            "fields": ["Comments"]
                        },
                        "CountryPost": {
                            "fields": ["Country", "PostalCode"]
                        },
                        "CustNum": {
                            "primary": true,
                            "unique": true,
                            "fields": ["CustNum"]
                        },
                        "Name": {
                            "fields": ["Name"]
                        },
                        "SalesRep2": {
                            "fields": ["SalesRep desc"]
                        }
                    },
                    "items": {
                        "additionalProperties": false,
                        "properties": {
                            "_id": {
                                "type": "string"
                            },
                            "CustNum": {
                                "type": "integer",
                                "ablType": "INTEGER",
                                "default": 0,
                                "title": "CustNum",
                                "required": true,
                                "readOnly": true
                            },
                            "Name": {
                                "type": "string",
                                "ablType": "CHARACTER",
                                "default": "",
```

```
                    "title": "Name"
                },
                "Address": {
                    "type": "string",
                    "ablType": "CHARACTER",
                    "default": "",
                    "title": "Address"
                },
                "Phone": {
                    "type": "string",
                    "ablType": "CHARACTER",
                    "default": "",
                    "title": "Phone"
                },
                "SalesRep": {
                    "type": "string",
                    "ablType": "CHARACTER",
                    "default": "",
                    "title": "SalesRep"
                },
                "Balance": {
                    "type": "number",
                    "ablType": "DECIMAL",
                    "default": 0,
                    "title": "Balance",
                    "width": 17,
                    "scale": 2
                },
                "State": {
                    "type": "string",
                    "ablType": "CHARACTER",
                    "default": "",
                    "title": "State"
                }
            }
        }
    }
},
```

PROPERTY	INFO
fields	Non-empty list of fields to be added to the index
primary	Optional, flag to indicate primary index
unique	Optional, flag to restrict indexed fields uniqueness
wordIndex	Optional, indicates full text index. For text fields only.

relations Property

The Relations property is optional. It should only be provided for DataSets with more than 1 table specified in the schema. It is an array that contains a “relations” entry for each relationship in the DataSet hierarchy. It is not a requirement that a table in a DataSet has to be related to another table, In fact, a DataSet can have multiple top-level tables, i.e. (a top-level table is a table that is not involved in a relationship as a child table).

```
"relations": [{
    "relationName": "<RelationName1>",
    "parentName"  : "<ParentName >",
    "childName"   : "<ChildName >",
    "relationFields": [{
        "parentFieldName": "<parentField1>",
        "childFieldName" : "<childField1>",
    },
    {
        "parentFieldName": "<parentField2>",
        "childFieldName" : "<childField2>"
    }]
},
{
    "relationName": "<RelationName2>",
    "parentName"  : "<ParentName >",
    "childName"   : "<ChildName >",
    "relationFields": [{
        "parentFieldName": "<parentField1>",
        "childFieldName" : "<childField1>",
    }]
}]
```

PROPERTY	INFO
relationName	Optional. Relation Name
parentName	Required. Name of parent table in relationship
childName	Required. Name of child table in relationship
relationFields	Required. Array of parent field name and child field name pairs. Can be used by the CDO to retrieve data for the child table based upon an equality match between all pairs of fields listed in this array.
parentFieldName	Required. Parent field for pair entry
childFieldName	Required. Child field for pair entry

Relations Example:

```
"relations": [
    {
        "relationName": "drel1",
        "parentName"  : "ttOrder",
        "childName"   : "ttOrderLine",
        "relationFields ": [{
            "parentFieldName": "OrderNum",
            "childFieldName" : "OrderNum"
        }]

    },
    {
        "parentName" : "ttOrderLine",
        "childName"  : "ttItem",
        "relationFields": [{
            "parentFieldName": "ItemNum",
            "childFieldName" : "ItemNum"
        }]
    }
]
```

dataDefinitions Property

The *dataDefinitions* property is available at the resource level. It includes schema definitions for DataSets, Tables, and Arrays of objects used by the invoke operations of the resource. It can be omitted from the catalog (or specified as an empty object) if there are no data definitions required by the resource's invoke operations.

A schema within the dataDefinitions property corresponds to one or more parameters in the operation's *params* propertythat have *xType* values set to either "DATASET", "TABLE" or "ARRAY". The name in the dataDefinitions section should correspond to the name of the data as it is specified in the params property for the invoke operation property.

```
"dataDefinitions": {
    {same as Schema definition – but without the properties property
        at the top}
    },
```

dataDefinitions Example:

```
"dataDefinitions": {
      "eOrder": {
            "type": "array",
            "additionalProperties": false,
            "properties": {
                "Ordernum": {
                    "type": "integer",
                    "ablType": "INTEGER",
                    "default": 0,
                    "title": "Order Num"
                },
                "CustNum": {
                    "type": "integer",
                    "ablType": "INTEGER",
                    "default": 0,
                    "title": "Cust Num"
                },
                "OrderDate": {
                    "type": "string",
                    "ablType": "DATE",
                    "default": null,
                    "title": "Ordered",
                    "format": "date"
                }
            }
        },
    }
```

operations Property

The Operations property is required. It is an array that provides information about each operation that the Mobile resource supports, information such as how the operation's parameters are mapped. This information from the catalog is used by the CDO to construct the HTTP request to be sent to the REST adapter. So, instead of a client app having to identify the URI, send an HTTP request, and interpret the HTTP response for a given REST resource operation call, it only has to call the appropriate method on the CDO to execute the corresponding operation on the resource.

The parameter information for each operation is specified in the operation's *params* property. The *params* property is optional. If an operation does not have any parameters, then it can be omitted. An empty *params* property can also be specified.

Note: In earlier catalog versions, the PATH and QUERY parameters were not listed in the *params* property. They were specified in the operation's *path* property.

```
"operations": [
    {
        "name": "<operation name1>",
        "path": "<relative uri>",
        "useBeforeImage": true | false,
        "type": "<method type>",
        "verb": "<verb type>",
        "mergeMode": "EMPTY" | "APPEND" | "MERGE" | "REPLACE",
        "params": [
            {
                "name": "<paramName1>",
                "type": "<paramType>"

            },
            {
                "name": "<paramName2>",
                "type": "<paramType>",
                "xType": <xtype value>,
                "isArray": true | false
            }
        ]
    },
    {
        "name": "<operation name2>",
        "path": "<relative uri>",
        "type": "<method type>",
        "verb": "<verb type>"
    }
]
```

OPERATIONS PROPERTY	INFO
Name	Optional. Operation Name
Path	Relative URI, run-time client info. Ex: /{CustNum} or ?Name={Name} Optional. If not specified, the CDO defaults to the resource level's path property. If specified, the operation's URL is the resource level's path property concatenated with the specified operation's path property. Ex. "/Customer" + "/{CustNum}" -> "/Customer/{CustNum} "
useBeforeImage	Optional. Default is: false.

<table>
<tr><td></td><td>Value should be either true or false. Specifies whether the data passed between the Business Entity and the CDO will be sent with before-image data</td></tr>
<tr><td>Type</td><td>Optional. Defaults to “invoke”.
<Method Type> can be either:
“create”
“read”
“update”
“delete”
“submit” (submit is a special invoke)
“invoke”

➢ create, read, update and delete correspond to built-in operations
➢ invoke corresponds to custom operations
➢ submit is a special invoke type</td></tr>
<tr><td>Verb</td><td><Verb Type> can be either:
“put”
“post”
“delete”
“get”

Optional. If not specified, CDO uses operation type to get corresponding default verb:

Operation Type — Default Verb
create — post
read — get
update, invoke, submit — put
delete — delete</td></tr>
<tr><td>mergeMode</td><td>Optional. This property tells the CDO to automatically merge to the CDO memory the data returned from an invoke operation. This property will be available at the operation level in the CDO catalog. The possible values for mergeMode correspond to the merge mode options for addRecords using idProperty as the key field:

• EMPTY: Empties the CDO memory and loads data.
• APPEND: Adds the data returned by the invoke operation to the existing data in the CDO memory. The method would throw an error if a duplicate key is found.
• MERGE: Merges the data returned by the invoke operation. If a duplicate key is found, the record will not be added to the CDO memory.
• REPLACE: Merges the data returned by the invoke operation. Records with a duplicate key are replaced.</td></tr>
</table>

Params	Optional. List of input and output parameters for the operation. Can be set to empty array ([]) to indicate that the operation does not have parameters.

PARAMS PROPERTY	INFO
Name	Required. Parameter name
Type	Required. Can be either: “PATH” “QUERY” “REQUEST_BODY” “RESPONSE_BODY”, “REQUEST_BODY, RESPONSE_BODY” (for input-output params) For V1.0 catalog, for CRUD operations, the PATH and QUERY parameters are only specified in the operation’s *path* property In the future, we may support: “MATRIX” “FORM” “COOKIE” “HEADER”
xType	Specifies parameter data type. Can be one of the following: “TABLE” “DATASET” “ARRAY” (for array of objects) json datatype: either “string”, “integer”, “number”, “boolean” ABL datatype: see OE ABL datatype mapping chart If “TABLE”, “DATASET”, or “ARRAY” is specified, then its schema needs to be specified in the resource’s **dataDefinitions** property.
isArray	True or False. Indicates whether parameter is an array of simple datatypes. For OE, this corresponds to the EXTENT option in the ABL

Samples

Sample 1: FamilyService.json file:

Sample catalog for resource “Family” defining CRUD operations and INVOKE operations. This is example of version 1.1 catalog format, where settings object is specified.

```
{
    "version": "1.1",
    "lastModified": "Mon Feb 24 14:07:51 EST 2014",
    "services": [{
        "name": "FamilyService",
        "address": "/rest/FamilyService",
        "settings": {
            "useRequest": true
        },
        "resources": [{
            "name": "Family",
            "path": "/Family",
            "schema": {
                "type": "object",
                "additionalProperties": false,
                "properties": {"dsFamily": {
                    "type": "object",
                    "additionalProperties": false,
                    "properties": {"ttFamily": {
                        "type": "array",
                        "items": {
                            "additionalProperties": false,
                            "properties": {
                                "_id": {"type": "string"},
                                "_errorString": {"type": "string"},
                                "BenefitDate": {
                                    "type": "string",
                                    "ablType": "DATE",
                                    "default": "2014-02-24",
                                    "title": "Benefit Date",
                                    "format": "date"
                                },
                                "Birthdate": {
                                    "type": "string",
                                    "ablType": "DATE",
                                    "default": "2014-02-24",
                                    "title": "Birthdate",
                                    "format": "date"
                                },
                                "CoveredOnBenefits": {
                                    "type": "boolean",
                                    "ablType": "LOGICAL",
```

```
                        "default": false,
                        "title": "Covered On Benefits"
                    },
                    "EmpNum": {
                        "type": "integer",
                        "ablType": "INTEGER",
                        "default": 0,
                        "title": "Emp No"
                    },
                    "RelativeName": {
                        "type": "string",
                        "ablType": "CHARACTER",
                        "default": "",
                        "title": "Relative Name"
                    }
                }
            }
        }}
    }}
},
"operations": [
    {
        "path": "",
        "useBeforeImage": false,
        "type": "create",
        "verb": "post",
        "params": [{
            "name": "dsFamily",
            "type": "REQUEST_BODY,RESPONSE_BODY"
        }]
    },
    {
        "name": "GetFamilyMember",
        "path": "/GetFamilyMember{piEmpNum}",
        "useBeforeImage": false,
        "type": "invoke",
        "verb": "put",
        "params": []
    },
    {
        "path": "?filter={filter}",
        "useBeforeImage": false,
        "type": "read",
        "verb": "get",
        "params": []
    },
    {
        "path": "",
        "useBeforeImage": false,
        "type": "update",
```

```
                "verb": "put",
                "params": [{
                    "name": "dsFamily",
                    "type": "REQUEST_BODY,RESPONSE_BODY"
                }]
            },
            {
                "path": "",
                "useBeforeImage": false,
                "type": "delete",
                "verb": "delete",
                "params": [{
                    "name": "dsFamily",
                    "type": "REQUEST_BODY,RESPONSE_BODY"
                }]
            },
            {
                "name": "GetFamilyCount",
                "path": "/GetFamilyCount",
                "useBeforeImage": false,
                "type": "invoke",
                "verb": "put",
                "params": [{
                    "name": "numMembers",
                    "xType": "INTEGER",
                    "type": "RESPONSE_BODY"
                }]
            }
        ]
    }]
}]
}
```

The following functions are available to the CDO based on this definition.
In the CDO syntax, “CDO” refers to the CDO and “family” refers to the JSRecord (an object for a specific record).

Operation	Verb	URL	CDO
CREATE	POST	/rest/FamilyService/Family	CDO.add([json-object])
READ	GET	/rest/FamilyService/Family?filter={filter}	CDO.fill([json-object])
UPDATE	PUT	/rest/FamilyService/Family	family.assign([json-object])
DELETE	DELETE	/rest/FamilyService/Family	family.remove( )
INVOKE	PUT	/rest/FamilyService/Family/GetFamilyMember/{piEmpNum}	CDO.GetFamilyMember([json-object])
INVOKE	PUT	/rest/FamilyService/Family/GetFamilyCount	CDO.GetFamilyCount( )

Notes:

- The URL is the concatenation of the service address + resource path + operation path.

- For the CUD operations (create, update, delete), the CDO performs the HTTP requests when executing the CDO.saveChanges() function.

Sample 2: CustomerStateSvc.json file:

Sample catalog for “CustomerStateSvc” service with 2 resources “Customer” and “State”. This is example of version 1.0 catalog format, where no settings object is specified.

```
{
    "version": "1.0",
    "lastModified": "Tue Feb 25 13:54:26 EST 2014",
    "services": [{
        "name": "CustomerStateSvc",
        "address": "/rest/CustomerStateSvc",
        "useRequest": true,
        "resources": [
            {
                "name": "State",
                "path": "/State",
                "schema": {
                    "type": "object",
                    "additionalProperties": false,
                    "properties": {"dsState": {
                        "type": "object",
                        "additionalProperties": false,
                        "properties": {"ttState": {
                            "type": "array",
                            "items": {
                                "additionalProperties": false,
                                "properties": {
                                    "_id": {"type": "string"},
                                    "_errorString": {"type": "string"},
                                    "State": {
                                        "type": "string",
                                        "ablType": "CHARACTER",
                                        "default": "",
                                        "title": "State"
                                    },
                                    "StateName": {
                                        "type": "string",
                                        "ablType": "CHARACTER",
                                        "default": "MA",
                                        "title": "State Name"
                                    },
```

```
                        "Region": {
                            "type": "string",
                            "ablType": "CHARACTER",
                            "default": "",
                            "title": "Region"
                        }
                    }
                }
            }}
        }},
        "operations": [
            {
                "path": "?filter={filter}",
                "useBeforeImage": false,
                "type": "read",
                "verb": "get",
                "params": []
            }
        ]
    },
    {
        "name": "Customer",
        "path": "/Customer",
        "schema": {
            "type": "object",
            "additionalProperties": false,
            "properties": {"dsCustomer": {
                "type": "object",
                "additionalProperties": false,
                "properties": {"ttCustomer": {
                    "type": "array",
                    "items": {
                        "additionalProperties": false,
                        "properties": {
                            "_id": {"type": "string"},
                            "_errorString": {"type": "string"},
                            "Address": {
                                "type": "string",
                                "ablType": "CHARACTER",
                                "default": "",
                                "title": "Address"
                            },
                            "Balance": {
                                "type": "number",
                                "ablType": "DECIMAL",
                                "default": 0,
                                "title": "Balance"
                            },
                            "City": {
```

```
                        "type": "string",
                        "ablType": "CHARACTER",
                        "default": "",
                        "title": "City"
                    },
                    "CustNum": {
                        "type": "integer",
                        "ablType": "INTEGER",
                        "default": 0,
                        "title": "Cust Num"
                    },
                    "Name": {
                        "type": "string",
                        "ablType": "CHARACTER",
                        "default": "",
                        "title": "Name"
                    },
                    "Phone": {
                        "type": "string",
                        "ablType": "CHARACTER",
                        "default": "",
                        "title": "Phone"
                    },
                    "PostalCode": {
                        "type": "string",
                        "ablType": "CHARACTER",
                        "default": "",
                        "title": "Postal Code"
                    }
                }
            }
        }}
    }}
},
"operations": [
    {
        "path": "",
        "useBeforeImage": false,
        "type": "update",
        "verb": "put",
        "params": [{
            "name": "dsCustomer",
            "type": "REQUEST_BODY"
        }]
    },
    {
        "path": "",
        "useBeforeImage": false,
        "type": "create",
        "verb": "post",
```

```
                    "params": [{
                        "name": "dsCustomer",
                        "type": "REQUEST_BODY"
                    }]
                },
                {
                    "path": "",
                    "useBeforeImage": false,
                    "type": "delete",
                    "verb": "delete",
                    "params": [{
                        "name": "dsCustomer",
                        "type": "REQUEST_BODY"
                    }]
                },
                {
                    "path": "?filter={filter}",
                    "useBeforeImage": false,
                    "type": "read",
                    "verb": "get",
                    "params": []
                }
            ]
        }
    ]
  }]
}
```

The following functions are available to the CDO based on this definition.
In the CDO syntax, “CDO” refers to the CDO and “customer” refers to the JSRecord (an object for a specific record).

Operation	**Verb**	**URL**	**CDO**
READ	GET	/rest/CustomerStateSvc /State?filter={filter}	CDO.fill([json-object])
CREATE	POST	/rest/CustomerStateSvc/Customer	CDO.add([json-object])
READ	GET	/rest/CustomerStateSvc /Customer?filter={filter}	CDO.fill([json-object])
UPDATE	PUT	/rest/CustomerStateSvc/Customer	customer.assign([json-object])
DELETE	DELETE	/rest/CustomerStateSvc/Customer	customer.remove( )

Notes:

- The URL is the concatenation of the service address + resource path + operation path-.
- For the CUD operations, the CDO performs the HTTP requests when executing the CDO.saveChanges() function.

Sample 3: DeptEmpServices.json file:

Sample catalog for an application with 2 services. “EmpMaintenanceSvc” contains the “Employee” resource. “DeptMaintenanceSvc” contains the “Department” resource.

```
{
    "version": "1.0",
    "lastModified": "Tue Feb 25 15:16:53 EST 2014",
    "services": [{
        "name": "EmpMaintenanceSvc",
        "address": "/rest/EmpMaintenanceSvc",
        "useRequest": true,
        "resources": [{
            "name": "Employee",
            "path": "/Employee",
            "schema": {
                "type": "object",
                "additionalProperties": false,
                "properties": {"dsEmployee": {
                    "type": "object",
                    "additionalProperties": false,
                    "properties": {"ttEmployee": {
                        "type": "array",
                        "items": {
                            "additionalProperties": false,
                            "properties": {
                                "_id": {"type": "string"},
                                "_errorString": {"type": "string"},
                                "Address": {
                                    "type": "string",
                                    "ablType": "CHARACTER",
                                    "default": "",
                                    "title": "Address"
                                },
                                "Address2": {
                                    "type": "string",
                                    "ablType": "CHARACTER",
                                    "default": "",
                                    "title": "Address2"
                                },
                                "Birthdate": {
                                    "type": "string",
                                    "ablType": "DATE",
                                    "default": "2014-02-25",
                                    "title": "Birthdate",
                                    "format": "date"
                                },
                                "City": {
                                    "type": "string",
                                    "ablType": "CHARACTER",
```

```
            "default": "",
            "title": "City"
        },
        "DeptCode": {
            "type": "string",
            "ablType": "CHARACTER",
            "default": "",
            "title": "Dept Code"
        },
        "EmpNum": {
            "type": "integer",
            "ablType": "INTEGER",
            "default": 0,
            "title": "Emp No"
        },
        "FirstName": {
            "type": "string",
            "ablType": "CHARACTER",
            "default": "",
            "title": "First Name"
        },
        "HomePhone": {
            "type": "string",
            "ablType": "CHARACTER",
            "default": "",
            "title": "Home Phone"
        },
        "LastName": {
            "type": "string",
            "ablType": "CHARACTER",
            "default": "",
            "title": "Last Name"
        },
        "Position": {
            "type": "string",
            "ablType": "CHARACTER",
            "default": "",
            "title": "Position"
        },
        "PostalCode": {
            "type": "string",
            "ablType": "CHARACTER",
            "default": "",
            "title": "Postal Code"
        },
        "SickDaysLeft": {
            "type": "integer",
            "ablType": "INTEGER",
            "default": 0,
            "title": "Sick Days Left"
```

```
                    },
                    "StartDate": {
                        "type": "string",
                        "ablType": "DATE",
                        "default": "2014-02-25",
                        "title": "Start Date",
                        "format": "date"
                    },
                    "State": {
                        "type": "string",
                        "ablType": "CHARACTER",
                        "default": "",
                        "title": "State"
                    },
                    "VacationDaysLeft": {
                        "type": "integer",
                        "ablType": "INTEGER",
                        "default": 0,
                        "title": "Vacation Days Left"
                    },
                    "WorkPhone": {
                        "type": "string",
                        "ablType": "CHARACTER",
                        "default": "",
                        "title": "Work Phone"
                    }
                }
            }
        }}
    }}
},
"operations": [
    {
        "path": "",
        "useBeforeImage": false,
        "type": "update",
        "verb": "put",
        "params": [{
            "name": "dsEmployee",
            "type": "REQUEST_BODY"
        }]
    },
    {
        "path": "",
        "useBeforeImage": false,
        "type": "delete",
        "verb": "delete",
        "params": [{
            "name": "dsEmployee",
            "type": "REQUEST_BODY"
```

```
                }]
            },
            {
                "path": "",
                "useBeforeImage": false,
                "type": "create",
                "verb": "post",
                "params": [{
                    "name": "dsEmployee",
                    "type": "REQUEST_BODY"
                }]
            },
            {
                "path": "?filter={filter}",
                "useBeforeImage": false,
                "type": "read",
                "verb": "get",
                "params": []
            }
        ]
    }]
},


   {
    "name": "DeptMaintenanceSvc",
    "address": "/rest/DeptMaintenanceSvc",
    "useRequest": true,
    "resources": [{
        "name": "Department",
        "path": "/Department",
        "schema": {
            "type": "object",
            "additionalProperties": false,
            "properties": {"dsDepartment": {
                "type": "object",
                "additionalProperties": false,
                "properties": {"ttDepartment": {
                    "type": "array",
                    "items": {
                        "additionalProperties": false,
                        "properties": {
                            "_id": {"type": "string"},
                            "_errorString": {"type": "string"},
                            "DeptCode": {
                                "type": "string",
                                "ablType": "CHARACTER",
                                "default": "",
                                "title": "Dept Code"
                            },
```

```
                    "DeptName": {
                        "type": "string",
                        "ablType": "CHARACTER",
                        "default": "",
                        "title": "Dept Name"
                    }
                }
            }
        }}
    }}
},
"operations": [
    {
        "path": "",
        "useBeforeImage": false,
        "type": "update",
        "verb": "put",
        "params": [{
            "name": "dsDepartment",
            "type": "REQUEST_BODY"
        }]
    },
    {
        "path": "?filter={filter}",
        "useBeforeImage": false,
        "type": "read",
        "verb": "get",
        "params": []
    },
    {
        "path": "",
        "useBeforeImage": false,
        "type": "delete",
        "verb": "delete",
        "params": [{
            "name": "dsDepartment",
            "type": "REQUEST_BODY"
        }]
    },
    {
        "path": "",
        "useBeforeImage": false,
        "type": "create",
        "verb": "post",
        "params": [{
            "name": "dsDepartment",
            "type": "REQUEST_BODY"
        }]
    }
]
```

```
        }]
    }

  ]
}
```

The following functions are available to the CDO based on this definition.
In the CDO syntax, “CDO” refers to the CDO and “emp” and “dept” refer to the JSRecord (an object for a specific record).

Operation	**Verb**	**URL**	**CDO**
CREATE	POST	/rest/EmpMaintenanceSvc/Employee	CDO.add([json-object])
READ	GET	/rest/EmpMaintenanceSvc/Employee ?filter={filter}	CDO.fill([json-object])
UPDATE	PUT	/rest/EmpMaintenanceSvc / Employee	emp.assign([json-object])
DELETE	DELETE	/rest/EmpMaintenanceSvc / Employee	emp.remove( )
CREATE	POST	/rest/DeptMaintenanceSvc /Department	CDO.add([json-object])
READ	GET	/rest/DeptMaintenanceSvc /Customer?filter={filter}	CDO.fill([json-object])
UPDATE	PUT	/rest/DeptMaintenanceSvc /Department	dept.assign([json-object])
DELETE	DELETE	/rest/ DeptMaintenanceSvc /Department	dept.remove( )

Notes:

- The URL is the concatenation of the service address + resource path + operation path-.
- For the CUD operations, the CDO performs the HTTP requests when executing the CDO.saveChanges() function.

Sample 4: SalesRepWithSubmitService.json file:

Sample catalog for the “SaleRep” resource with “submit” operation.

```
{
    "version": "1.0",
    "lastModified": "Thu Feb 27 09:56:30 EST 2014",
    "services": [{
        "name": "SalesRepService",
        "address": "/rest/SalesRepService",
        "useRequest": true,
        "resources": [{
            "name": "SaleRep",
            "path": "/SaleRep",
            "schema": {
                "type": "object",
                "additionalProperties": false,
                "properties": {"dsSalesrep": {
                    "type": "object",
                    "additionalProperties": false,
                    "properties": {"ttSalesrep": {
                        "type": "array",
                        "items": {
                            "additionalProperties": false,
                            "properties": {
                                "_id": {"type": "string"},
                                "_errorString": {"type": "string"},
                                "MonthQuota": {
                                    "type": "array",
                                    "maxItems": 12,
                                    "items": {"type": "integer"},
                                    "ablType": "INTEGER",
                                    "default": 0
                                },
                                "Region": {
                                    "type": "string",
                                    "ablType": "CHARACTER",
                                    "default": "",
                                    "title": "Region"
                                },
                                "RepName": {
                                    "type": "string",
                                    "ablType": "CHARACTER",
                                    "default": "",
                                    "title": "Rep Name"
                                },
                                "SalesRep": {
                                    "type": "string",
                                    "ablType": "CHARACTER",
                                    "default": "",
```

```
                        "title": "Sales Rep"
                    }
                }
            }
        }}
    }},
},
"operations": [
    {
        "path": "?filter={filter}",
        "useBeforeImage": true,
        "type": "read",
        "verb": "get",
        "params": []
    },
    {
        "path": "",
        "useBeforeImage": true,
        "type": "update",
        "verb": "put",
        "params": [{
            "name": "dsSalesrep",
            "type": "REQUEST_BODY"
        }]
    },
    {
        "name": "GetData",
        "path": "/GetData",
        "useBeforeImage": true,
        "type": "invoke",
        "verb": "put",
        "params": [{
            "name": "inparam1",
            "type": "REQUEST_BODY"
        }]
    },
    {
        "name": "SubmitSaleRep",
        "path": "/SubmitSaleRep",
        "useBeforeImage": true,
        "type": "submit",
        "verb": "put",
        "params": [{
            "name": "dsSalesrep",
            "type": "REQUEST_BODY"
        }]
    },
    {
        "path": "",
        "useBeforeImage": true,
```

```
                    "type": "create",
                    "verb": "post",
                    "params": [{
                        "name": "dsSalesrep",
                        "type": "REQUEST_BODY"
                    }]
                },
                {
                    "path": "",
                    "useBeforeImage": true,
                    "type": "delete",
                    "verb": "delete",
                    "params": [{
                        "name": "dsSalesrep",
                        "type": "REQUEST_BODY"
                    }]
                }
            ]
        }]
    }]
}
```

The following functions are available to the CDO based on this definition.
In the CDO syntax, “CDO” refers to the CDO and “salesrep” refers to the JSRecord (an object for a specific record).

Operation	**Verb**	**URL**	**CDO**
CREATE	POST	/rest/SalesRepService/SalesRep	CDO.add([json-object])
READ	GET	/rest/SalesRepService/SalesRep?filter={filter}	CDO.fill([json-object])
UPDATE	PUT	/rest/ SalesRepService/SalesRep	salesrep.assign([json-object])
SUBMIT	PUT	/rest/SalesRepService/SalesRep/SubmitSalesRep	CDO.saveChanges(true)
DELETE	DELETE	/rest/ SalesRepService/SalesRep	salesrep.remove( )
INVOKE	PUT	/rest/SalesRepService/SalesRep/GetData	CDO.GetData([json-object])

Notes:

- The URL is the concatenation of the service address + resource path + operation path-.
- For the CUD operations (create, update, delete), the CDO performs the HTTP requests when executing the CDO.saveChanges() or CDO.saveChanges(false).
- For the submit operation, the CDO performs a single HTTP request when executing the CDO.saveChanges(true) function.

Sample 5: rb_catalog.json

Sample catalog with both input and output parameters.

```
{
  "version": "1.3",
  "lastModified": "5/11/15 11:05 AM",
  "services": [ {
       "name": "RB_contact_Service",
       "address": "https://www.rollbase.com/rest/",
       "settings": {
              "useRequest": false,
              "sendOnlyChanges": true,
              "unwrapped": false,
              "useXClientProps": true
       },
       "resources": [ {
              "name": "contact",
              "path": "mobile/api",
              "displayName": "Contact",
              "idProperty": "id",
              "schema": {
                    "type": "object",
                    "additionalProperties": false,
                    "properties": {
                        "contact": {
                           "type": "array",
                           "items": {
                               "additionalProperties" : false,
                               "properties" : {
                                  "_id": {
                                      "type": "string",
                                      "default": "",
                                      "title": "_id"
                                  },
                                  "Raccount": {
                                      "type": "string",
                                      "title": "Account"
                                  },
                                  "assistant": {
                                      "type": "string",
                                      "title": "Assistant"
                                  },
                                  "asst__phone": {
                                      "type": "string",
                                      "title": "Asst. Phone"
                                  },
                                  "birthdate": {
                                      "type": "string",
                                      "title": "Birthdate",
                                      "format": "date"
                                  },
                                  "Ccontact": {
                                      "type": "string",
```

```
                    "title": "Child Contacts"
                },
                "city": {
                    "type": "string",
                    "title": "City"
                },
                "name": {
                    "type": "string",
                    "title": "Contact"
                },
                "contactOwner": {
                    "type": "string",
                    "title": "Contact Owner"
                },
                "country": {
                    "type": "number",
                    "title": "Country"
                },
                "firstName": {
                    "type": "string",
                    "title": "First Name"
                },
                "home_phone": {
                    "type": "string",
                    "title": "Home Phone"
                },
                "id": {
                    "type": "number",
                    "title": "ID"
                },
                "job_title": {
                    "type": "string",
                    "title": "Job Title"
                },
                "lead_source": {
                    "type": "number",
                    "title": "Lead Source"
                },

                "state": {
                    "type": "number",
                    "title": "State/Province"
                },
                "streetAddr": {
                    "type": "string",
                    "title": "Street Address 1"
                },
                "zip": {
                    "type": "string",
                    "title": "ZIP/Postal Code"
                }
            }
        }
    }
}
```

```
},
"dataDefinitions": {
       "picklistData": {
             "type": "array",
             "properties": {
                 "code": {
                     "type": "string",
                    "default": "",
                    "title": "Code"
                 },
                 "id": {
                   "type": "integer",
                   "default": "-1",
                   "title": "Id"
                 },
                 "name": {
                   "type": "string",
                   "default": "",
                   "title": "Name"
                 }
             }
       },
       "genericData" : {
             "type" : "array",
             "properties" : {
                 "id" : {
                   "type" : "integer",
                   "default" : "-1",
                   "title" : "ID"
                 },
                 "name" : {
                   "type" : "string",
                   "default": "",
                   "title": "Name"
                 }
             }
      }
},
"operations": [
    {
        "name": "getPicklist_country",
         "path":
            "/getPicklist?output=json&objDefId=107440343&fieldDefId=109144675",
         "type": "Invoke",
         "verb": "get",
         "params": [ {
             "name": "picklistData",
             "xType": "ARRAY",
             "type": "RESPONSE_BODY"
         }]
    },


    {
         "name": "getPicklist_salutation",
```

```
        "path":
           "/getPicklist?output=json&objDefId=107440343&fieldDefId=109144696",
        "type": "Invoke",
        "verb": "get",
        "params" : [ {
             "name": "picklistData",
            "xType": "ARRAY",
            "type": "RESPONSE_BODY"
        }]
    },
    {
        "name": "Create",
        "path": "/create2?output=json&useIds=true&objName=contact",
        "type": "Create",
        "verb": "post",
        "params": [
            {
                "name": "contact",
                "type": "REQUEST_BODY",
                "xType": "DATASET"
            },
            {
                "name": "contact",
                "type": "RESPONSE_BODY",
                "xType": "DATASET"
            }]
     },
     {
        "name": "Read",
        "path": "/getDataObj?output=json&id={filter}&objName=contact",
        "type": "Read",
        "verb": "get",
        "params": [ {
            "name": "contact",
            "type": "RESPONSE_BODY",
            "xType": "DATASET"
        }]
     },
     {
        "name" : "Update",
        "path" : "/update2?output=json&useIds=true&objName=contact",
        "type" : "Update",
        "verb" : "put",
        "params" : [
            {
                "name" : "contact",
                "type" : "REQUEST_BODY",
                "xType" : "DATASET"
            },
            {
                "name" : "contact",
                "type" : "RESPONSE_BODY",
                "xType" : "DATASET"
             }]
      },
```

```
            {
                "name": "Delete",
                "path": "/delete?output=json&id={id}&objName=contact",
                "type": "Delete",
                "verb": "delete",
                "params": [
                    {
                        "name": "contact",
                        "type": "REQUEST_BODY",
                        "xType": "DATASET"
                    },
                    {
                        "name": "contact",
                        "type": "RESPONSE_BODY",
                        "xType": "DATASET"
                    }]
            },
            {
                "name": "getView_All_Contacts",
                "path":
"/getPage?output=json&startRow={startRow}&rowsPerPage={rowsPerPage}&viewId=107444236&
objName=contact",
                "type": "Invoke",
                "verb": "get",
                "mergeMode": "EMPTY",
                "params": [{
                    "name": "contact",
                    "type": "RESPONSE_BODY",
                    "xType": "DATASET"
                 }]
            },
            {
                "name" : "getAll_RB188961",
                "path" :
"/getRelationships?output=json&id={id}&startRow={startRow}&rowsPerPage={rowsPerPage}&
relName=RB188961&fieldList=id,name",
                "type" : "Invoke",
                "verb" : "get",
                "params" : [ {
                    "name" : "genericData",
                    "xType" : "ARRAY",
                    "type" : "RESPONSE_BODY"
                 }]
            }
        ]
    }]
  }]
}
```

Sample 6: Schema for DataSet with nested Data-Relation

Support was added for DataSets defined with nested Data-Relations. The format of the catalog file is the same for nested and non-nested DataSets except for the schema property. This example shows the schema property in a catalog for the following ABL DataSet:

```
DEFINE DATASET dsCustomerOrder
    FOR eCustomer, eOrder
    DATA-RELATION custOrdRel FOR eCustomer, eOrder
        RELATION-FIELDS (CustNum, CustNum) NESTED.
```

The JSON data that will be sent/returned to/from the server will be in nested format. A property referencing the child table is added to the parent table's property list. Note that in this format, the child table (eOrder) is defined as an "object "instead of an "array".

```
"schema": {
       "type": "object",
        "additionalProperties": false,
        "properties": {
              "dsCustomerOrder": {
                    "type": "object",
                       "additionalProperties": false,
                       "properties": {
                            "eCustomer": {
                                "type": "array",
                                "items": {
                                    "additionalProperties": false,
                                    "properties": {
                                        "_id": {"type": "string"},
                                        "CustNum": {
                                            "type": "integer",
                                            "ablType": "INTEGER",
                                            "default": 0,
                                            "title": "CustNum"
                                        },
                                        "Name": {
                                            "type": "string",
                                            "ablType": "CHARACTER",
                                            "default": "",
                                            "title": "Name"
                                        },
                                        "eOrder": {
                                            "type": "array",
                                            "ablType": "null",
                                            "items": {"$ref":
"#/properties/dsCustomerOrder/properties/eOrder"}
                                        }
```

```
                                    }
                                }
                            },
                            "eOrder": {
                               "type": "object",
                               "additionalProperties": false,
                                "properties": {
                                    "_id": {"type": "string"},
                                    "Ordernum": {
                                        "type": "integer",
                                        "ablType": "INTEGER",
                                        "default": 0,
                                        "title": "Ordernum"
                                    },
                                    "CustNum": {
                                        "type": "integer",
                                        "ablType": "INTEGER",
                                        "default": 0,
                                        "title": "CustNum"
                                    },
                                    "OrderDate": {
                                        "type": "string",
                                        "ablType": "DATE",
                                        "default": null,
                                        "title": "OrderDate",
                                        "format": "date"
                                    }
                                }
                            }
                        }
                }
        }
},
"relations": [{
        "relationName": "custOrdRel",
        "parentName": "eCustomer",
        "childName" : "eOrder",
        "relationFields": [{
             "parentFieldName": "CustNum",
             "childFieldName" : "CustNum"
        }]
}],
```

Sample 7: Schema for DataSet with 2 nested Data-Relations

This example shows the schema property in a catalog for the following DataSet with 3 levels:

```
DEFINE DATASET dsDeptEmpFamily
    FOR eDepartment, eEmployee, eFamily
    DATA-RELATION deptEmpRel FOR eDepartment, eEmployee
        RELATION-FIELDS (DeptCode, DeptCode) NESTED

    DATA-RELATION empFamilyRel FOR eEmployee, eFamily
        RELATION-FIELDS (EmpNum, EmpNum) NESTED.
```

Note that eEmployee and eFamily are designated as “object”s, and have a reference to them in another table. eDepartment is designated as an “array”, and is not referenced in another table.

```
"schema": {
    "type": "object",
    "additionalProperties": false,
    "properties": {
             "dsDeptEmpFamily": {
                   "type": "object",
                     "additionalProperties": false,
                     "properties": {
                           "eDepartment": {
                               "type": "array",
                               "items": {
                                   "additionalProperties": false,
                                   "properties": {
                                       "_id": {"type": "string"},
                                       "DeptCode": {
                                           "type": "string",
                                           "ablType": "CHARACTER",
                                           "default": "",
                                           "title": "DeptCode"
                                       },
                                       "DeptName": {
                                           "type": "string",
                                           "ablType": "CHARACTER",
                                           "default": "",
                                           "title": "DeptName"
                                       },
                                       "eEmployee": {
                                           "type": "array",
                                           "ablType": "null",
                                           "items":{"$ref":
    "#/properties/dsDeptEmpFamily/properties/eEmployee"}
                                       }
                                   }
                               }
```

```
  },
"eEmployee": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
          "_id": {"type": "string"},
          "EmpNum": {
              "type": "integer",
              "ablType": "INTEGER",
              "default": 0,
              "title": "EmpNum"
          },
          "LastName": {
              "type": "string",
              "ablType": "CHARACTER",
              "default": "",
              "title": "LastName"
          },
          "FirstName": {
              "type": "string",
              "ablType": "CHARACTER",
              "default": "",
              "title": "FirstName"
          },
           "DeptCode": {
              "type": "string",
              "ablType": "CHARACTER",
              "default": "",
              "title": "DeptCode"
          },
          "Position": {
              "type": "string",
              "ablType": "CHARACTER",
              "default": "",
              "title": "Position"
          },

           "eFamily": {
              "type": "array",
              "ablType": "null",
              "items":{"$ref":
"#/properties/dsDeptEmpFamily/properties/eFamily"}
          }
      }
  },
"eFamily": {
      "type": "object",
      "additionalProperties": false,
```

```
                                "properties": {
                                    "_id": {"type": "string"},
                                    "EmpNum": {
                                        "type": "integer",
                                        "ablType": "INTEGER",
                                        "default": 0,
                                        "title": "EmpNum"
                                    },
                                    "RelativeName": {
                                        "type": "string",
                                        "ablType": "CHARACTER",
                                        "default": "",
                                        "title": "RelativeName"
                                    },
                                    "Relation": {
                                        "type": "string",
                                        "ablType": "CHARACTER",
                                        "default": "",
                                        "title": "Relation"
                                    },
                                    "CoveredOnBenefits": {
                                        "type": "boolean",
                                        "ablType": "LOGICAL",
                                        "default": false,
                                        "title": "CoveredOnBenefits"
                                    }
                                }
                            }
                        }
                    }
                }
            },


"relations": [{
    "relationName": "deptEmpRel",
    "parentName": "eDepartment",
    "childName": "eEmployee",
    "relationFields": [{
             "parentFieldName": "DeptCode",
             "childFieldName" : "DeptCode"
    }]
},
{
    "relationName": "empFamilyRel",
    "parentName": "eEmployee",
    "childName": "eFamily",
```

```
    "relationFields": [{
             "parentFieldName": "EmpNum",
             "childFieldName": "EmpNum"
    }]
}],
```

Sample 8: Schema for DataSet with 2 Data-Relations, nested & not

This example shows the schema property in a catalog for a DataSet called dsCustomerOrderNestedState with 2 data-relations. The relationship for eCustomer and eOrder is nested. The relationship for eCustomer and eState is not nested.

```
DEFINE DATASET dsCustomerOrderNestedState
    FOR eCustomer, eOrder, eState
    DATA-RELATION custOrdRel FOR eCustomer, eOrder
        RELATION-FIELDS (CustNum, CustNum) NESTED
    DATA-RELATION custStateRel FOR eCustomer, eState
        RELATION-FIELDS (State, State).
```

Note that eOrder is designated as an “object”, and has a reference to it in eCustomer. eCustomer and eState are designated as “array”s, and are not referenced in another table.

```
"schema": {
       "type": "object",
        "additionalProperties": false,
        "properties": {
             "dsCustomerOrderNestedState": {
                 "type": "object",
                  "additionalProperties": false,
                  "properties": {
                       "eCustomer": {
                            "type": "array",
                            "items": {
                                "additionalProperties": false,
                                "properties": {
                                    "_id": {"type": "string"},
                                    "CustNum": {
                                        "type": "integer",
                                        "ablType": "INTEGER",
                                        "default": 0,
                                        "title": "CustNum"
                                    },
                                    "Name": {
                                        "type": "string",
                                        "ablType": "CHARACTER",
                                        "default": "",
                                        "title": "Name"
                                    },
                                    "State": {
                                        "type": "string",
                                        "ablType": "CHARACTER",
                                        "default": "",
                                        "title": "State"
                                    },
                                    "eOrder": {
                                        "type": "array",
                                        "ablType": "null",
```

```
                                    "items": {"$ref":
"#/properties/dsCustomerOrder/properties/eOrder"}
                                }
                            }
                        }
                    },
                    "eOrder": {
                        "type": "object",
                        "additionalProperties": false,
                        "properties": {
                            "_id": {"type": "string"},
                            "Ordernum": {
                                "type": "integer",
                                "ablType": "INTEGER",
                                "default": 0,
                                "title": "Ordernum"
                            },
                            "CustNum": {
                                "type": "integer",
                                "ablType": "INTEGER",
                                "default": 0,
                                "title": "CustNum"
                            },
                            "OrderStatus": {
                                "type": "string",
                                "ablType": "CHARACTER",
                                "default": "Ordered",
                                "title": "OrderStatus"
                            }
                        }
                    },
                    "eState": {
                        "type": "array",
                        "items": {
                            "additionalProperties": false,
                            "properties": {
                                "_id": {"type": "string"},
                                "State": {
                                    "type": "string",
                                    "ablType": "CHARACTER",
                                    "default": "",
                                    "title": "State"
                                },
                                "StateName": {
                                    "type": "string",
                                    "ablType": "CHARACTER",
                                    "default": "",
                                    "title": "StateName"
                                },
                                "Region": {
                                    "type": "string",
                                    "ablType": "CHARACTER",
                                    "default": "",
                                    "title": "Region"
```

```
                                    }
                                }
                            }
                        }
                    }
                }
        }
},

"relations": [{
        "relationName": "custOrdRel",
                "parentName": "eCustomer",
                "childName": "eOrder",
                "relationFields": [{
                        "parentFieldName": "CustNum",
                        "childFieldName": "CustNum"
                }]
},
{
        "relationName": "custStateRel",
                "parentName": "eCustomer",
                "childName": "eState",
                "relationFields": [{
                        "parentFieldName": "State",
                        "childFieldName": "State"
                }]
}],
```

Sample 9: Schema for DataSet with multiple tables (with only 2 data-relations)

```
DEFINE DATASET dsCustomerExtraTables
    FOR ttCustomer, ttOrder, ttOrderLine, ttState, ttSalesRep
    DATA-RELATION custOrdRel FOR ttCustomer, ttOrder
        RELATION-FIELDS (CustNum, CustNum)
    DATA-RELATION OrdOrdLineRel FOR ttOrder, ttOrderLine
        RELATION-FIELDS (Ordernum, Ordernum).

"schema": {
    "type": "object",
    "additionalProperties": false,
    "properties": {
      "dsCustomerExtraTables": {
        "type": "object",
        "additionalProperties": false,
        "properties": {
            "ttCustomer": {
                "type": "array",
                "items": {
                    "additionalProperties": false,
                    "properties": {
                        "_id": {"type": "string"},
                        "_errorString": {"type": "string"},
                        "Address": {
                            "type": "string",
                            "ablType": "CHARACTER",
                            "default": "",
                            "title": "Address"
                        },
                        "Balance": {
                            "type": "number",
                            "ablType": "DECIMAL",
                            "default": 0,
                            "title": "Balance"
                        },
                        "City": {
                            "type": "string",
                            "ablType": "CHARACTER",
                            "default": "",
                            "title": "City"
                        },
                        "CustNum": {
                            "type": "integer",
                            "ablType": "INTEGER",
                            "default": 0,
                            "title": "Cust Num"
                        },
                        "Name": {
                            "type": "string",
                            "ablType": "CHARACTER",
```

```
                    "default": "",
                    "title": "Name"
                },
                "Phone": {
                    "type": "string",
                    "ablType": "CHARACTER",
                    "default": "",
                    "title": "Phone"
                },
                "State": {
                    "type": "string",
                    "ablType": "CHARACTER",
                    "default": "",
                    "title": "State"
                }
            }
        }
    },
    "ttOrder": {
        "type": "array",
        "items": {
            "additionalProperties": false,
            "properties": {
                "_id": {"type": "string"},
                "_errorString": {"type": "string"},
                "Ordernum": {
                    "type": "integer",
                    "ablType": "INTEGER",
                    "default": 0,
                    "title": "Ordernum"
                },
                "CustNum": {
                    "type": "integer",
                    "ablType": "INTEGER",
                    "default": 0,
                    "title": "CustNum"
                },
                "OrderDate": {
                    "type": "string",
                    "ablType": "DATE",
                    "default": null,
                    "title": "OrderDate",
                    "format": "date"
                },
                "SalesRep": {
                    "type": "string",
                    "ablType": "CHARACTER",
                    "default": "",
                    "title": "SalesRep"
                },
                "OrderStatus": {
                    "type": "string",
                    "ablType": "CHARACTER",
                    "default": "Ordered",
                    "title": "OrderStatus"
```

```
            }
        }
    }
},
"ttOrderline": {
    "type": "array",
    "items": {
        "additionalProperties": false,
        "properties": {
            "_id": {"type": "string"},
            "_errorString": {"type": "string"},
            "Ordernum": {
                "type": "integer",
                "ablType": "INTEGER",
                "default": 0,
                "title": "Ordernum"
            },
            "Linenum": {
                "type": "integer",
                "ablType": "INTEGER",
                "default": 0,
                "title": "Linenum"
            },
            "Itemnum": {
                "type": "integer",
                "ablType": "INTEGER",
                "default": 0,
                "title": "Itemnum"
            },
            "Price": {
                "type": "number",
                "ablType": "DECIMAL",
                "default": 0,
                "title": "Price"
            },
            "Qty": {
                "type": "integer",
                "ablType": "INTEGER",
                "default": 0,
                "title": "Qty"
            }
        }
    }
},
"ttState": {
    "type": "array",
    "items": {
        "additionalProperties": false,
        "properties": {
            "_id": {"type": "string"},
            "_errorString": {"type": "string"},
            "State": {
                "type": "string",
                "ablType": "CHARACTER",
                "default": "",
```

```
                        "title": "State"
                    },
                    "StateName": {
                        "type": "string",
                        "ablType": "CHARACTER",
                        "default": "",
                        "title": "StateName"
                    },
                    "Region": {
                        "type": "string",
                        "ablType": "CHARACTER",
                        "default": "",
                        "title": "Region"
                    }
                }
            }
        },
        "ttSalesRep": {
            "type": "array",
            "items": {
                "additionalProperties": false,
                "properties": {
                    "_id": {"type": "string"},
                    "_errorString": {"type": "string"},
                    "SalesRep": {
                        "type": "string",
                        "ablType": "CHARACTER",
                        "default": "",
                        "title": "SalesRep"
                    },
                    "RepName": {
                        "type": "string",
                        "ablType": "CHARACTER",
                        "default": "",
                        "title": "RepName"
                    },
                    "Region": {
                        "type": "string",
                        "ablType": "CHARACTER",
                        "default": "",
                        "title": "Region"
                    },
                    "MonthQuota": {
                        "type": "array",
                        "maxItems": 12,
                        "items": {"type": "integer"},
                        "ablType": "INTEGER",
                        "default": 0
                    }
                }
            }
        }
    }
},
```