

JSONRPC: Extending the datastore

Up to and including Oxygen, an external entity which would like to present some data as a datastore to ODL via JSON RPC must support the following operations. These map onto JSON RPC methods with the same name in a single RPC endpoint representing the data. While there is a facility to spread a "data store" image over multiple endpoints it has not been tested up to and including oxygen.

- The "Entity" in the API description is roughly equivalent to a device under management. We deliberately do not use device as a term here as an entity can merge multiple devices into a single view or be one of the many management endpoints within a larger device.
- Path up to and including Oxygen is path as described in the yang modelled JSON-RPC draft. We are presently discussing the possibility to replace it a path defined using the RFC8040 grammar.

Non Transactional

read

Read a piece of yang modelled data via JSON-RPC

Arguments	Entity, Datastore, Path
Returns	JSON Data

Unfortunately, ODL does not support the notion of transactional read. You cannot perform a read on a half-finished and uncommitted transaction sequence. You can only see committed data. This is the reason why read does not take a transaction id argument.

exists

Check if data exists identified by Entity, Datastore, Path via JSON-RPC

Arguments	Entity, Datastore, Path
Returns	boolean

Unfortunately, ODL does not support the notion of transactional exist. You cannot perform an exist on a half-finished and uncommitted transaction sequence. You can only see committed data. This is the reason why exists does not take a transaction id argument.

txid

Obtain an identifier to be used for a sequence of transactional operations.

Arguments	None
Returns	String to uniquely identify a transaction for any of the transaction operations.

All existing implementations use the string representation of a type 4 UUID for this operation. This is not mandatory, any unique string will work and be accepted as ODL.

error

Obtain detailed error information (usually for a failed transaction).

Arguments	Transaction ID - a string provided by the txid function to identify a transaction.
Returns	List of detailed application specific error codes which provides information in addition to the normal JSON RPC error codes.

Transactional

put

Enqueue an unconditional creation of the supplied data element at the supplied path (and all necessary parent elements leading to it). If the element exists - enqueue an overwrite to it. Associate this request with this transaction. The actual operation is not performed until a commit is issued for this transaction.

Arguments	Transaction ID, Entity, Datastore, Path, Data
Returns	Nothing

merge

Enqueue a modification of an element using the supplied data at the supplied path. If the element does not exist - fail. Associate this request with this transaction. The actual operation is not performed until a commit is issued for this transaction.

Arguments	Transaction ID, Entity, Datastore, Path, Data
Returns	Nothing

delete

Enqueue a deletion of an element using the supplied data at the supplied path. Associate this request with this transaction. The actual operation is not performed until a commit is issued for this transaction.

Arguments	Transaction ID, Entity, Datastore, Path
Returns	Nothing

commit

Commit all operations associated with this transaction.

Arguments	Transaction ID
Returns	True if successful, False if unsuccessful.

If unsuccessful, the server implementation is obliged to rollback and clean-up.

cancel

Cancel all operations associated with this transaction.

Arguments	Transaction ID
Returns	True if successful, False if unsuccessful.

The implementation is obliged to rollback and clean-up.

add_listener

Add a data change listener for a path.

The server should allocate at least one URI for notifications. When the change is triggered a notification at a given URI with the given name will be produced. The combination URI+ DCN Name should be unique across all entities and datastores supported by a particular endpoint.

The URI may be reused across multiple listeners. In that case the actual triggered listener is identified by the DCN Name.

The notification may contain the changed tree data from the path specified in the DCN creation request downwards or null.

The actual choice - to add or not the actual changed content to the notification is up to the implementer on the server side. Client developers may rely that the payload if not null is the changed data from the DCN request path downwards, but should not rely on the notification being always non-null. If the payload body is null, the client implementation may need to query the data to fetch the changes.

If a portion of the tree which has listeners associated with it is deleted all notifications are triggered with a null payload. There is unfortunately no way to distinguish between a notification for a deletion and a notification for which the server has opted not to provide "modified data" payload.

DCNs located with a part of the tree which is deleted fire only at the moment of deletion. From there onwards they are orphaned and will not fire again. Re-creation of the tree at a later date does not trigger a re-association of orphaned DCNs to the newly created tree.

Caller may specify transport protocol which will be used to communicate data changes. When omitted, implementation will use same protocol as was used in this request.

Arguments	Entity, Datastore, Path, Transport
Returns	URI, DCN Name.

Notes:

- A good choice for "DCN Notification Name" is a type 4 UUID in textual representation.
- It is up to the server implementation to decide if it wants to reuse notification endpoints and service two different requesters which have requested the same path using the same notification URI + "DCN Notification Name"

delete_listener

Delete data change listener

Arguments	URI, DCN Name
Returns	True if successful, False otherwise.

Future

These operations are work in progress and are in fact proposals for future functionality

invoke_rpc

Invoke RPC method

Arguments	Method name, RPC input
Returns	RPC output.

publish_notification

Publish notification

Arguments	Notification name, Payload
Returns	N/A.