RESTful Network API for Twinning Devices

Candidate Version 1.0 – 15 Dec 2015

Open Mobile Alliance

OMA-TS-REST_NetAPI-Twinning-V1_0-20151215-C

Contents

Figures

Tables

1. Scope

This specification defines a RESTful API for Twinning Devices using HTTP protocol bindings.

2. References

2.1 Normative References

[Autho4API_10]	"Authorization Framework for Network APIs", Open Mobile Alliance™, OMA-ER-Autho4API-V1_0, URL:http://www.openmobilealliance.org/
[REST_NetAPI_ACR]	"RESTful Network API for Anonymous Customer Reference Management", Open Mobile Alliance™, OMA-TS-REST_NetAPI_ACR-V1_0, URL:http://www.openmobilealliance.org/
[REST_NetAPI_Common]	"Common definitions for RESTful Network APIs", Open Mobile Alliance™, OMA-TS-REST_NetAPI_Common-V1_0, URL:http://www.openmobilealliance.org/
[REST_NetAPI_NotificationChannel]	"RESTful Network API for Notification Channel", Open Mobile Alliance™, OMA-TS-REST_NetAPI_NotificationChannel-V1_0, URL:http://www.openmobilealliance.org/
[REST_SUP_Twinning]	"XML schema for the RESTful Network API for Twinning Devices", Open Mobile Alliance™, OMA-SUP-XSD_rest_netapi_twinning-V1_0, URL:http://www.openmobilealliance.org/
[RFC2119]	"Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL:http://tools.ietf.org/html/rfc2119
[RFC3966]	"The tel URI for Telephone Numbers", H.Schulzrinne, December 2004, URL:http://tools.ietf.org/html/rfc3966
[RFC3986]	"Uniform Resource Identifier (URI): Generic Syntax", R. Fielding et. al, January 2005, URL:http://tools.ietf.org/html/rfc3986
[RFC7159]	"The JavaScript Object Notation (JSON) Data Interchange Format", T. Bray, Ed., March 2014, URL:http://tools.ietf.org/html/rfc7159
[RFC7231]	"Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", R. Fielding, J. Reschke, June 2014, URL: http://tools.ietf.org/html/rfc7231.txt
[SCRRULES]	"SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://www.openmobilealliance.org/
[XMLSchema1]	W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures Second Edition, W3C Recommendation 5 April 2012, URL:http://www.w3.org/TR/xmlschema11-1/
[XMLSchema2]	W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes, W3C Recommendation 5 April 2012, URL:http://www.w3.org/TR/xmlschema11-2/

2.2 Informative References

[OMADICT]	"Dictionary for OMA Specifications", Version 2.8, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_9, URL:http://www.openmobilealliance.org/
[REST_NetAPI_Messaging]	"RESTful Network API for Messaging", Open Mobile Alliance™, OMA-TS-REST_NetAPI_Messaging-V1_0, URL:http://www.openmobilealliance.org/
[REST_NetAPI_NMS]	"RESTful Network API for Network Message Storage", Open Mobile Alliance™, OMA-TS-REST_NetAPI_NMS-V1_0, URL:http://www.openmobilealliance.org/
[REST_WP]	"Guidelines for RESTful Network APIs", Open Mobile Alliance™, OMA-WP-Guidelines_for_RESTful_Network_APIs, URL:http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

For the purpose of this TS, all definitions from the OMA Dictionary apply [OMADICT].

Client-side Notification URL	An HTTP URL exposed by a client, on which it is capable of receiving notifications and that can be used by the client when subscribing to notifications.
Long Polling	A variation of the traditional polling technique, where the server does not reply to a request unless a particular event, status or timeout has occurred. Once the server has sent a response, it closes the connection, and typically the client immediately sends a new request. This allows the emulation of an information push from a server to a client.
Notification Channel	A channel created on the request of the client and used to deliver notifications from a server to a client. The channel is represented as a resource and provides means for the server to post notifications and for the client to receive them via specified delivery mechanisms. For example in the case of Long Polling the channel resource is defined by a pair of URLs. One of the URLs is used by the client as a call-back URL when subscribing for notifications. The other URL is used by the client to retrieve notifications from the Notification Server.
Notification Server	A server that is capable of creating and maintaining Notification Channels.
Primary Device	A device in a Twinning relationship which shares its identity with another device referred to as the Secondary Device.
Secondary Device	A device in a Twinning relationship which assumes the identity of the Primary Device.
Server-side Notification URL	An HTTP URL exposed by a Notification Server, that identifies a Notification Channel and that can be used by a client when subscribing to notifications.
Twinning	The network provisioning process in which the identity of a device (Primary Device) is shared with another device (Secondary Device). Such a process would result in a Twinning relationship between the two devices in which the Secondary Device assumes the identity of the Primary Device.

3.3 Abbreviations

ACR	Anonymous Customer Reference
API	Application Programming Interface
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
MCC	Mobile Country Code
MIME	Multipurpose Internet Mail Extensions
MNC	Mobile Network Code
MNO	Mobile Network Operator
OMA	Open Mobile Alliance

REST	REpresentational State Transfer
SCR	Static Conformance Requirements
SIP	Session Initiation Protocol
TS	Technical Specification
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WP	White Paper
XML	eXtensible Markup Language
XSD	XML Schema Definition

4. Introduction

The Technical Specification of the RESTful Network API for Twinning Devices contains HTTP protocol bindings for Twinning functionality, using the REST architectural style. The specification provides resource definitions, the HTTP verbs applicable for each of these resources, and the element data structures, as well as support material including flow diagrams and examples using the various supported message body formats (i.e. XML and JSON).

4.1 Version 1.0

Version 1.0 of this specification supports the following operations. These operations also support the inter-MNO Twinning interactions between two MNOs (i.e. use cases involving Primary and Secondary Devices served by different Mobile Network Operators).

- Activate a Twinning relationship between a Primary and a Secondary Device
- Retrieve the list of devices which a given device is in Twinning relationship with
- Request a Twinning “Activation Code”
- Toggle Twinning status On/Off
- Deactivate a Twinning relationship
- Retrieve information for a given Twinning relationship
- Retrieve status of a Twinning relationship
- Manage (create, refresh, delete) subscriptions to notifications related to Twinning relationship events
- Notify changes in the Twinning relationship status (On/Off or deactivation)
- Manage inter-MNO Twinning relationships (used in MNO-to-MNO interactions)
- Acquire an inter-MNO Twinning “Activation Code” (used in MNO-to-MNO interactions)
- Update user preferences parameters of a Twinning relationship (e.g. Twinning relationship’s name)

In addition, this specification provides:

- Support for scope values used with authorization framework defined in [Autho4API_10]
- Support for Anonymous Customer Reference (ACR) as an end user identifier
- Support for “acr:auth” as a reserved keyword in an ACR

5. Twinning Devices API definition

This section is organized to support a comprehensive understanding of the Twinning Devices API design. It specifies the definition of all resources, definition of all data structures, and definitions of all operations permitted on the specified resources.

Common data types, naming conventions, fault definitions and namespaces are defined in [REST_NetAPI_Common].

The remainder of this document is structured as follows:

Section 5 starts with a description of the concepts used by this API (section 5.1). This is followed by a diagram representing the resources hierarchy followed by a table listing all the resources (and their URL) used by this API, along with the data structure and the supported HTTP verbs (section5.2). What follows are the data structures (section 5.3). A sample of typical use cases is included in section 5.4, described as high level flow diagrams.

Section 6 contains detailed specification for each of the resources. Each such subsection defines the resource, the request URL variables that are common for all HTTP methods, and the supported HTTP verbs. For each supported HTTP verb, a description of the functionality is provided, along with an example of a request and an example of a response. For each unsupported HTTP verb, the returned HTTP error status is specified, as well as what should be returned in the Allow header.

All examples in section 6 use XML as the format for the message body, while JSON examples are provided in Appendix D.

Section 7 contains fault definition details such as Service Exceptions and Policy Exceptions.

Appendix B provides the Static Conformance Requirements (SCR).

Appendix C provides application/x-www-form-urlencoded examples, where applicable.

Appendix E provides the operations mapping to a pre-existing baseline specification, where applicable.

Appendix F provides a list of all Light-weight Resources, where applicable.

Appendix G defines authorization aspects to control access to the resources defined in this specification.

Note: Throughout this document client and application can be used interchangeably.

5.1 Concepts

5.1.1 Twinning relationship

A Twinning relationship consists of two devices where one takes on the Primary role and the other takes on the role of the Secondary Device. As a result of the Twinning relationship activation, the Secondary Device would assume the identity of the Primary Device. Twinning activation would result in the Secondary Device being able to receive voice calls and messages destined for the Primary Device, originate calls/messages and be perceived by the destination as the Primary Device.

A Secondary Device MAY form a Twinning relationship with only one Primary Device. However, a Primary Device MAY form Twinning relationship with several Secondary Devices.

If a Primary Device has several Twinning relationships then, its role towards all of its Twinning relationships SHALL be Primary.

As a result of the Twinning activation, a Secondary Device SHALL have access to all the Primary Device’s services as authorized by the Primary Device (e.g. as defined by the Primary Device’s OAuth access_token provided to the Secondary Device as part of the Twinning activation). Whether a Secondary Device can take advantage of the Primary Device’s authorized services is dependent on the Secondary Device’s capability as well as the Secondary Device MNO’s network capabilities.

5.1.2 Secondary Device interactions with Primary Device’s network

Once the Twinning activation is in place, the Secondary Device, depending upon its inherent capabilities, is able to send/receive messages in addition to place/receive calls on behalf of the Primary Device.

In order to send/receive messages, the Secondary Device can directly interface with the Primary Device’s network using appropriate Network APIs (e.g. [REST_NetAPI_Messaging] and [REST_NetAPI_NMS]). For this purpose the Primary Device’s MNO should provide an appropriate OAuth access token to the Secondary Device. For further information see “authCode” parameter in section 5.3.2.5.

The Secondary Device can also place voice calls on behalf of the Primary Device (i.e. be perceived by the destination endpoint as if the calls are originated from the Primary Device) and also receive voice calls which are destined for the Primary Device. However, for such voice call related functionalities, the Secondary Device does not need to communicate with the Primary Device’s network over Network APIs (hence, there is no need for the Secondary Device to receive an OAuth access token from the Primary Device).

5.1.3 Inter-MNO Twinning

In an inter-MNO Twinning scenario, the Primary and Secondary Devices are managed by different MNOs. As a result, Twinning operations (e.g. Twining activation, Twinning deactivation and Twinning Status Toggle) which are initiated by one Twinning device (e.g. Primary) in one network is expected to be relayed by the respective MNO server to the other device’s (e.g. Secondary) MNO server. Such interactions are supported by the resources defined in section 5.2 of this specification.

5.2 Resources Summary

This section summarizes all the resources used by the RESTful Network API for Twinning Devices.

The "apiVersion" URL variable SHALL have the value "v1" to indicate that the API corresponds to this version of the specification. See [REST_NetAPI_Common] which specifies the semantics of this variable.

The figure below visualizes the resource structure defined by this specification. Note that those nodes in the resource tree which have associated HTTP methods defined in this specification are depicted by solid boxes.

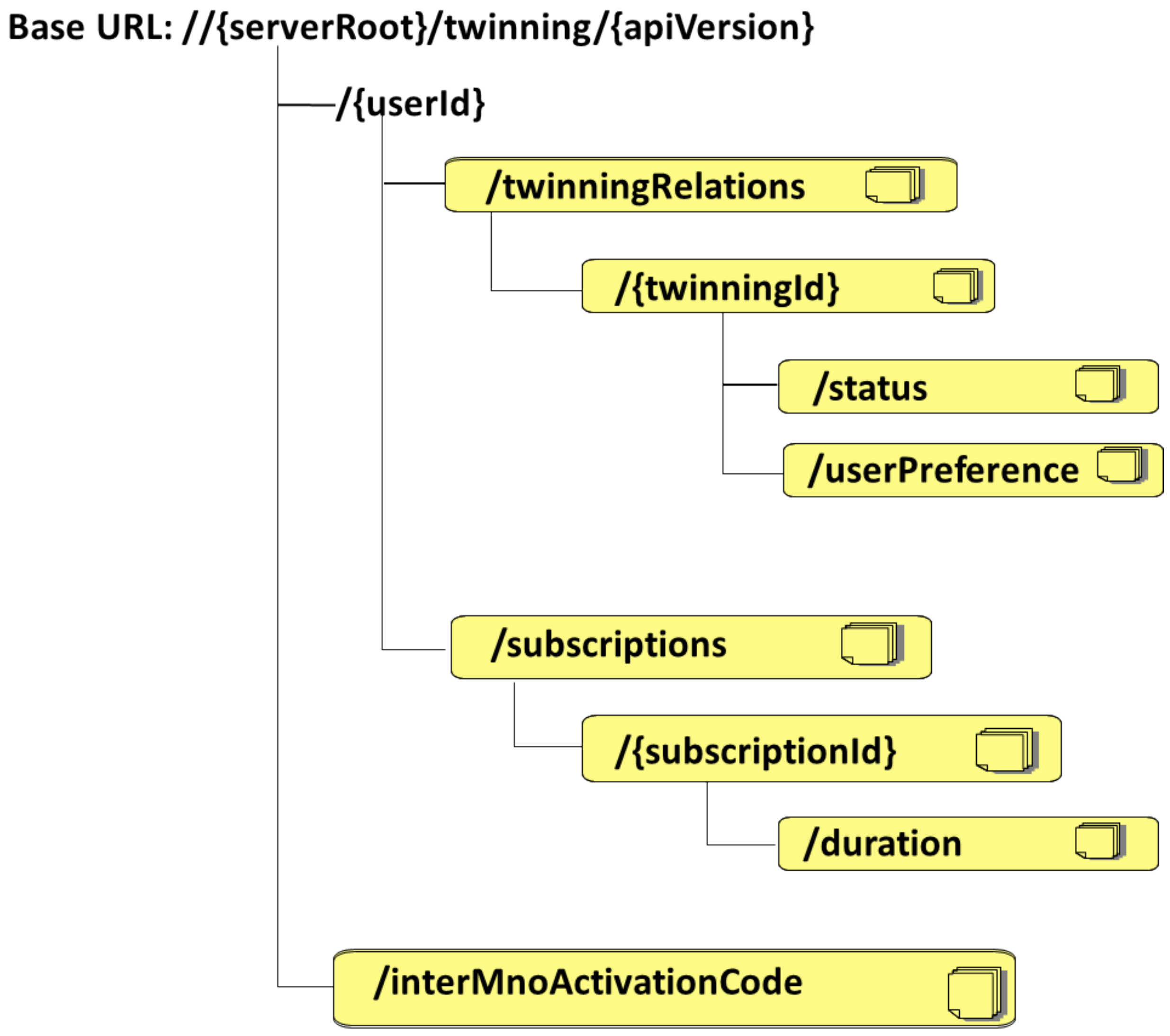

Figure 1 Resource structure defined by this specification

The following tables give a detailed overview of the resources defined in this specification, the data type of their representation and the allowed HTTP methods.

5.2.1 To allow a client to manage Twinning Relationship

Resource	**URL** **Base URL: //{serverRoot}/twinning/{apiVersion}/{userId}**	**Data Structures**	**HTTP verbs**			
			GET	**PUT**	**POST**	**DELETE**
Twinning relationships	/twinningRelations	TwinningList (used for GET) Twinning (used for POST) common:ResourceReference (optional alternative for POST response)	Retrieves the existing Twinning relationships for the end user's device identified by the {userId}. Note: Query string parameter used to provide Twinning activation instruction and MNO list.	no	Creates a new Twinning relationship	no
An individual Twinning relationship	/twinningRelations/{twinningId}	Twinning (used for GET and POST Response) InterMnoTwinningActivation (used for POST Request)	Retrieves Twinning data	no	Activates an inter-MNO "pending" Twinning relationship (specifically used in inter-MNO use cases)	Removes a Twinning relationship
Twinning status	/twinningRelations/{twinningId}/status	Status	Retrieves Twinning status	Toggles Twinning status (to "On/Off")	no	no
Twinning user preference parameters	/twinningRelations/{twinningId}/userPreference	UserPreference	no	Updates Twinning user preferences parameters (e.g. Twinning name)	no	no

5.2.2 To allow a client to manage subscriptions to notifications related to Twinning events

Resource	**URL** **Base URL:** **//{serverRoot}/twinning/{apiVersion}/{userId}**	**Data Structures**	**HTTP verbs**			
			GET	**PUT**	**POST**	**DELETE**
All subscriptions to Twinning notifications	/subscriptions	TwinningSubscriptionList (used for GET) TwinningSubscription (used for POST) common:ResourceReference (optional alternative for POST response)	Retrieves all active Twinning notification subscriptions	no	Creates new subscription for notification for Twinning status changes	no
Individual subscription to Twinning notifications	/subscriptions/{subscriptionId}	TwinningSubscription (used for GET response)	Retrieves an individual subscription	no	no	Cancels subscription and stop corresponding notifications
Subscription duration	/subscriptions/{subscriptionId}/duration	SubscriptionDuration	no	Extends (“refreshes”) the duration of a subscription	no	no

5.2.3 To allow an MNO to interact with another MNO to request an Activation code

Resource	**URL** **Base URL:** **//{serverRoot}/twinning/{apiVersion}/**	**Data Structures**	**HTTP verbs**			
			GET	**PUT**	**POST**	**DELETE**
Inter-MNO Twinning activation code	/interMnoActivationCode	InterMnoActivationCode	no	no	Requests a new Twinning Activation Code	no

5.2.4 Resources allowing the server to notify a client about Twinning events

Resource	URL <specified by the client>	Data Structures	HTTP verbs			
			GET	PUT	POST	DELETE
Client notification about Twinning events	<specified by the client when subscription is created or during provisioning process>	TwinningEventNotification	no	no	Notifies client about Twinning events	no

5.3 Data Types

5.3.1 XML Namespaces

The XML namespace for the Twinning Devices data types is:

urn:oma:xml:rest:netapi:twinning:1

The 'xsd' namespace prefix is used in the present document to refer to the XML Schema data types defined in XML Schema [XMLSchema1, XMLSchema2]. The 'common' namespace prefix is used in the present document to refer to the data types defined in [REST_NetAPI_Common]. The use of namespace prefixes such as 'xsd' is not semantically significant.

The XML schema for the data structures defined in the section below is given in [REST_SUP_Twinning].

5.3.2 Structures

The subsections of this section define the data structures used in the Twinning Devices API.

Some of the structures can be instantiated as so-called root elements.

For structures that contain elements which describe a user identifier, the statements in section 6 regarding 'tel', 'sip' and 'acr' URI schemes apply.

5.3.2.1 Type: TwinningSubscriptionList

This type represents a list of subscriptions to notifications regarding Twinning events.

Element	Type	Optional	Description
subscription	TwinningSubscription[0..unbounded]	Yes	Array of notification subscriptions.
resourceURL	xsd:anyURI	No	Self referring URL.

A root element named twinningSubscriptionList of type TwinningSubscriptionList is allowed in response bodies.

5.3.2.2 Type: TwinningSubscription

This type represents a subscription to notifications regarding Twinning events.

Element	Type	Optional	Description
callbackReference	common:CallbackReference	No	Client's Notification URL and OPTIONAL callbackData.
duration	xsd:unsignedInt	Yes	Period of time (in seconds) notifications are provided for. If set to “0” (zero), or omitted, a duration time will be chosen according to service provider policy. This element MAY be given by the client during resource creation in order to signal the desired lifetime of the subscription. The server MUST return in this element the period of time for which the subscription will still be valid.

clientCorrelator	xsd:string	Yes	A correlator that the client can use to tag this particular resource representation during a request to create a resource on the server. This element MAY be present. Note: this allows the client to recover from communication failures during resource creation and therefore avoids duplicate subscriptions in such situations. In case the element is present, the server SHALL not alter its value, and SHALL provide it as part of the representation of this resource. In case the field is not present, the server SHALL NOT generate it.
resourceURL	xsd:anyURI	Yes	Self referring URL. The resourceURL SHALL NOT be included in POST requests by the client, but MUST be included in POST requests representing notifications by the server to the client, when a complete representation of the resource is embedded in the notification. The resourceURL MUST also be included in responses to any HTTP method that returns an entity body, and in PUT requests.

A root element named twinningSubscription of type TwinningSubscription is allowed in request and/or response bodies.

5.3.2.3 Type: SubscriptionDuration

Duration of a subscription to Twinning notifications.

Element	Type	Optional	Description
duration	xsd:unsignedInt	Yes	Period of time (in seconds) notifications are provided for. If set to “0” (zero), or omitted (not present) in PUT requests, a default duration time will be chosen according to service provider policy. This element can be specified by the client in PUT requests when refreshing a subscription however, the server in the response to the request may change the desired lifetime according to its server policy. The server MUST always include the subscription duration (lifetime) in the response.

A root element named subscriptionDuration of type SubscriptionDuration is allowed in request and/or response bodies.

5.3.2.4 Type: TwinningList

This type represents a list of Twinning relationships.

Element	Type	Optional	Description
twinning	Twinning[0..unbounded]	Yes	List of Twinning relationships.
instruction	Instruction	Yes	General Twinning activation instruction and the list of available MNOs which a Twinning device (e.g. Secondary) may establish a Twinning relationship with. Query parameter "Instruction" MAY be used by the application to request this element's inclusion in the response from the server.
resourceURL	xsd:anyURI	No	Self referring URL.

A root element named twinningList of type TwinningList is allowed in response bodies.

5.3.2.5 Type: Twinning

This type represents a Twinning relationship.

Element	Type	Optional	Description
role	TwinningRole	Yes	The role ("Primary" or "Secondary") of this Twinning device as provided in the request (POST) body during resource creation request. This parameter is needed by the server in Twinning resource creation operation. If "role" is not provided by the client in the request, it is assumed that the sever has other means of retrieving the role information (e.g. through OAuth token present in the request or in advance pre-configuration information at the server, etc.). If the client does not provide the "role" as part of the request and the server has no way of determining the intended "role" through other means, an appropriate error SHALL be resulted (see sections 6.1.5.8 and 7.1.1)
primaryMno	Mno	Yes	This element MAY be present in the request (POST) body during resource creation only if role = Secondary. • If role = Secondary, the value of primaryMno determines which server (i.e. MNO) is to generate the "activationCode". If primaryMno is different than Secondary's MNO, the server SHALL interact with the Primary's MNO server to obtain an "activationCode". This results in an inter-MNO interaction between the two MNO's (see interMnoActivationCode resource in section 5.2 for further information). • If role = Secondary and this element is not

			present the server SHALL assume the Primary Device's MNO is the same as Secondary's. • If role = Primary, this element is not relevant and MUST be absent (if present, it SHALL be ignored by the server).
name	xsd:string	Yes	Human readable name (e.g. myCar or myWatch) of the Twinning relationship which SHOULD be provided by the client (e.g. by Primary's device) in the request (POST) body during resource creation If name is not provided by the client, the server SHALL assign a unique name (e.g. car1, wearable1). The server SHALL ensure the name is unique within the context of a given Primary Device.
otherDevice	OtherDevice	Yes	Information pertaining to the other device in this Twinning relationship. This element SHALL be populated by the server upon successful Twinning activation. It SHALL NOT be present in the request (POST) body during resource creation. In inter-MNO scenarios (where the primary and Secondary Devices are managed by different MNOs), Primary Device's server SHALL use a POST request to update the Secondary Device's Twinning resource with information pertaining to the Primary Device. (see sections 5.3.2.6, 5.4.7and 6.2.5).
activationCode	TwinningActivationCode	Yes	A single-use Twinning "activationCode". This element MUST be present in POST requests only if role = Primary. If role = Secondary, this element is not relevant and MUST be absent from the POST requests (if present, it SHALL be ignored by the server). The "activationCode" SHALL be generated by the server and MUST be provided in the response to the resource creation request (POST) when role=Secondary. Once the Twinning relationship is activated successfully, the "activationCode" SHALL be invalidated by the server (i.e. it MAY no longer be used in a subsequent resource creation requests (POST) by the Primary Device).
twinningStatus	TwinningStatus	Yes	The status of the Twinning relationship. The "twinningStatus" SHALL NOT be present in the POST request body during resource creation. It SHALL be set by the server and returned in the

			response. Secondary Device's twinningStatus SHALL by default be set to "Pending" by the server upon creation of the Twinning resource (i.e. due to POST request). Primary Device's twinningStatus SHALL be set to "On" by the server upon successful activation of the Twinning resource on the Primary Device's side. The twinningStatus of the Secondary Device SHALL be updated to twinningStatus = On by Primary Device's server upon successful Twinning relationship activation on the Primary's side. In inter-MNO scenarios (where the primary and Secondary Devices are managed by different MNOs), Primary Device's server SHALL use a POST request to update the Secondary Device's Twinning resource which effectively updates twinningStatus accordingly. See sections 5.3.2.6, 5.4.7 and 6.2.5 for further information.
authCode	xsd:string	Yes	This element contains an OAuth authorization code which SHALL be generated by the Primary Device's server and populated in the Secondary Device's Twinning resource (i.e. {twinningId}) upon successful activation of the Twinning relationship. It SHALL NOT be present in the POST request body during resource creation. In an inter-MNO scenario (where the primary and secondary participants are managed by different MNOs), Primary Device's server SHALL use a POST request to populate the Secondary Device's Twinning resource accordingly. See section 5.3.2.6, 5.4.7 and 6.2.5 for further information. "authCode" is to be used by the Secondary Device to communicate with the Primary Device's Authorization server (e.g. OAuth 2.0) and obtain an appropriate OAuth Access Token (and possibly Refresh Token) which enables it to interact with the Primary's network enablers (e.g. messaging enabler in order to send/receive messages on behalf of the Primary Device). See [Autho4API_10] for further details on obtaining OAuth Access Token using Authorization Code flow. This element is not relevant when role=Primary.
primaryDeviceVMAddress	xsd:anyURI	Yes	The Primary Device's Voice Mail retrieval access address (e.g. 'sip' URI, 'tel' URI, 'acr' URI). This element SHALL be populated by the server upon successful Twinning activation. It SHALL NOT be present in the POST request body during resource creation. In inter-MNO scenarios (where the primary and Secondary Devices are managed by different MNOs), Primary Device's server SHALL use a POST request to update the Secondary Device's resource with

			information pertaining to Primary Device's Voice Mail retrieval access address along with other necessary parameters. See section 5.3.2.6 for further information.
outOfBandNotifyPrimaryUser	xsd:boolean	Yes	This element MAY be present in the request (POST) body during resource creation only if role = Primary. If set to true, the server SHALL inform the Primary user of Twinning events via other 'out of band' means (e.g. SMS, email, etc.). If set to false, then based on server's policy, the server either SHALL NOT send any 'out of band' Twinning events or limit the Twinning events which are sent 'out of band' (e.g. stop toggle on/off events but still send "Deactivated" event) to the Primary user. The default value (i.e. when this parameter is absent), is false. If role = Secondary, this element is not relevant and MUST be absent (if present, it SHALL be ignored by the server). Note: Notifying Primary user 'out of band', of Twinning events is independent of direct Twinning event notifications which may be sent to the Primary Device's Twinning application client (through event subscription).
outOfBandNotifyPrimaryUserId	xsd:anyURI	Yes	The endpoint at which the user would prefer to receive out of band (e.g. SMS) Twinning status change notifications (see outOfBandNotifyPrimaryUser element for further information). The server's policy MAY limit the value of this element to a set of known/verifiable values (e.g. Primary's MSISDN, Primary user's email address). In that case, attempting to set the value of this element to any value other than the permitted values SHALL result in a Policy error. The default value (i.e. when this parameter is absent), is Primary's MSISDN. If role = Secondary, this element is not relevant and MUST be absent (if present, it SHALL be ignored by the server).
ownAddress	xsd:anyURI	Yes	Self referring address (e.g. 'sip' URI, 'tel' URI, 'acr' URI) of this device. This element SHALL be populated by the server. It SHALL NOT be present in the POST request body during resource creation.
resourceURL	xsd:anyURI	Yes	Self referring URL. The resourceURL SHALL NOT be included in POST requests by the client, but MUST be included in POST requests representing notifications by the server to the

			client, when a complete representation of the resource is embedded in the notification. The resourceURL MUST also be included in responses to any HTTP method that returns an entity body, and in PUT requests.

A root element named twinning of type Twinning is allowed in request and/or response bodies.

5.3.2.6 Type: InterMnoTwinningActivation

This type represents an inter-MNO Twinning activation (See section 6.2.5 for further information).

Element	Type	Optional	Description
name	xsd:string	No	Human readable name (e.g. myCar or myWatch) of the Twinning relationship.
otherDevice	OtherDevice	No	Information pertaining to Primary Device in this Twinning relationship.
authCode	xsd:string	Yes	This element contains an OAuth authorization code. If present, it SHALL be generated by the Primary Device's server and populated in the Secondary Device's Twinning resource (i.e. {twinningId}) upon successful activation of the Twinning relationship. authCode is to be used by the Secondary Device to communicate with the Primary Device's Authorization server (e.g. OAuth 2.0) and obtain an appropriate OAuth Access Token (and possibly Refresh Token) which enables it to interact with the Primary's network enablers (e.g. messaging enabler and network message store enabler in order to send/receive messages on behalf of the Primary Device). See [Autho4API_10] for further details on obtaining OAuth Access Token using Authorization Code flow.
primaryDeviceVMAddress	xsd:anyURI	Yes	The Primary Device's Voice Mail retrieval access address (e.g. 'sip' URI, 'tel' URI, 'acr' URI).

A root element named interMnoTwinningActivationof type InterMnoTwinningActivationis allowed in request bodies.

5.3.2.7 Type: OtherDevice

This type represents the other device in a given Twinning relationship.

Element	Type	Optional	Description
resourceURL	xsd:anyURI	No	The other device's Twinning resourceURL (i.e. ../{twinningId}) in this Twinning relationship.
address	xsd:anyURI	No	The address (e.g. 'sip' URI, 'tel' URI, 'acr' URI) of the other device in this Twinning relationship.
mno	Mno	No	The MNO serving the other device in this Twinning relationship.

5.3.2.8 Type: InterMnoActivationCode

This type represents an inter-MNO Twinning activation code associated with a Twinning resource managed by another MNO.

Element	Type	Optional	Description
twinningResourceURL	xsd:anyURI	No	The resourceURL of the Twinning resource on the requesting MNO side which is enquiring a Twinning activationCode.
address	xsd:anyURI	No	The address (e.g. 'sip' URI, 'tel' URI, 'acr' URI) of the Twinning device on the requesting MNO side which is enquiring a Twinning activationCode.
activationCode	TwinningActivationCode	Yes	A single-use Twinning “activationCode”. It SHALL NOT be present in the POST request. The “activationCode” SHALL be generated by the server and MUST be provided in the response to the POST request.

A root element named interMnoActivationCode of type InterMnoActivationCode is allowed in request and/or response bodies.

5.3.2.9 Type: TwinningActivationCode

This type represents a Twinning Activation Code.

Element	Type	Optional	Description
code	xsd:string	No	Single-use Twinning activation code.
mno	Mno	Yes	Identifying the MNO which generated the activation “code”. This element MUST be populated by the server and included in server’s response.

5.3.2.10 Type: Instruction

This type represents Twinning activation instructions.

Element	Type	Optional	Description
mno	Mno[0..unbounded]	Yes	List of MNOs which a Twinning device (e.g. Secondary) may form a Twinning relationship with.
activationInstruction	xsd:string	Yes	Instruction which MAY be used by an end user to know what are the steps in performing a Twinning activation between two devices.

5.3.2.11 Type: Mno

This type represents an MNO's identity

Element	Type	Optional	Description
name	xsd:string	No	MNO name.
ncc	xsd:string	No	Network code (ncc) is the concatenation MCC and MNC. An example ncc of "23415" consists of the concatenation of MCC="234" and MNC="15".

5.3.2.12 Type: UserPreference

This type represents the Twinning relationship's user preference parameters.

Element	Type	Optional	Description
name	xsd:string	Yes	Human readable name (e.g. myCar or myWatch) of the Twinning relationship. Attempting to set the name of a Twinning relationship to a value which has already been used for another Twinning relationship in the context of a given Primary Device SHALL result in an appropriate error response (see section 6.4.4.5). In PUT requests, if name is not provided by the client, the server SHALL assign a unique name (e.g. car1, wearable1).The server SHALL ensure the name is unique within the context of a given Primary Device.
outOfBandNotifyPrimaryUser	xsd:boolean	Yes	If set to true, the server SHALL inform the Primary user of Twinning events via other 'out of band' means (e.g. SMS, email, etc.). If set to false, then based on server's policy, the server either SHALL NOT send any 'out of band' Twinning events or limit the Twinning events which are sent 'out of band' (e.g. stop toggle on/off events but still send "Deactivated" event) to the Primary user. The default value (i.e. when this parameter is absent), is false.
outOfBandNotifyPrimaryUserId	xsd:anyURI	Yes	The endpoint at which the user would prefer to receive out of band (e.g. SMS) Twinning status change notifications (see outOfBandNotifyPrimaryUser element in section 5.3.2.5 for further information). The server's policy MAY limit the value of this element to a set of known/verifiable values (e.g. Primary's MSISDN, Primary user's email address). In that case, attempting to set the value of this element to any value other than the permitted values SHALL result in a Policy error. The default value (i.e. when this parameter is absent), is Primary's MSISDN.

A root element named userPreference of type UserPreference is allowed in request and/or response bodies.

5.3.2.13 Type: Status

This type represents the Twinning status.

Element	Type	Optional	Description
twinningStatus	TwinningStatus	No	The Twinning status.

A root element named status of type Status is allowed in request and/or response bodies.

5.3.2.14 Type: TwinningEventNotification

This type represents a notification about a Twinning event.

Element	Type	Optional	Description
callbackData	xsd:string	Yes	The ‘callbackData’ element if it was passed by the application in the ‘callbackReference’ element when creating a subscription to notifications about Twinning events. See [REST_NetAPI_Common]
eventType	TwinningEventType	No	Type of event
twinningResourceURL	xsd:anyURI	No	The resource URL of the Twinning object. This can be used by the client application to retrieve further information about the Twinning relationship (e.g. “otherDevice”, “authCode”, etc).
link	Common:Link [0..unbounded]	Yes	Link to other resources that are in relationship with this notification. The server SHOULD include a link to the related subscription. No other links are required or suggested by this specification.

A root element named twinningEventNotification of type TwinningEventNotification is allowed in notification request bodies.

5.3.3 Enumerations

The subsections of this section define the enumerations used in the Twinning Devices API.

5.3.3.1 Enumeration: TwinningRole

This enumeration defines the Twinning role of a device.

Enumeration	Description
Primary	Primary Device in a Twinning relationship.
Secondary	Secondary Device in a Twinning relationship.

5.3.3.2 Enumeration: TwinningStatus

This enumeration defines the Twinning Status of a Twinning resource (i.e. {twinningId}).

Enumeration	Description
On	Twinning status is on.
Off	Twinning status is off.
Pending	Twinning status is pending. It SHALL be set by the server only.

TwinningStatus values in the request (e.g. PUT) body represent the desired status of the Twinning resource upon successful completion of the operation. In the response it represents the current status of Twinning resource.

An operation can be accepted for processing but the processing has not been completed, i.e. the Twinning resource is not in its final status. In such a case the Server MAY return an HTTP 202 Accepted status code, along with a representation of the resource, in response to the request. In this response the TwinningStatus value SHALL be “Pending”. The application client can later check the twinning status using a GET operation on the resource or receive a notification of the final status of the twinning resource once the execution of the request is complete.

The support of the HTTP 202 Accepted response is OPTIONAL for the server although the application client MUST support it. If the server supports the HTTP 202 Accepted response, the server SHALL also support GET operations on {twinningId} and/or “status” resource and SHOULD support notification of final status of twinning resource.

5.3.3.3 Enumeration: TwinningEventType

This enumeration defines the types of events. It is used in notifications.

Enumeration	Description
Activated	Twinning relationship is activated. When a pending twinning relationship becomes active (i.e. TwinningStatus is changed from Pending to On) an event notification with this event type is emitted.
Deactivated	Twinning relationship is permanently removed (i.e. deactivated). When a Twinning relationship is removed, an event notification with this event type is emitted.
ToggledOn	Twinning relationship is toggled on (i.e. TwinningStatus is set to On). When a Twinning relationship is toggled on, an event notification with this event type is emitted.
ToggledOff	Twinning relationship is toggled off (i.e. TwinningStatus is set to Off). When a Twinning relationship is toggled off, an event notification with this event type is emitted.

5.3.4 Values of the Link “rel” attribute

The “rel” attribute of the Link element is a free string set by the server implementation, to indicate a relationship between the current resource and an external resource. The following are possible strings (list is non-exhaustive, and can be extended):

- TwinningSubscription

These values indicate the kind of resource that the link points to.

5.4 Sequence Diagrams

The following subsections describe the resources, methods and steps involved in typical scenarios.

In a sequence diagram, a step which involves delivering a notification is labeled with “POST or NOTIFY”, where “POST” refers to delivery via the HTTP POST method, and “NOTIFY” refers to delivery using the Notification Channel [REST_NetAPI_NotificationChannel].

In the following sequence diagrams, the actor lines “Primary Device”, “Secondary Device” or “Application” refer to the corresponding Twining Application executing on the Primary or the Secondary Device.

5.4.1 Subscription to Twinning notifications

This figure below shows a scenario for an application subscribing to Twinning notifications (i.e. status change), querying for a list of active subscriptions, querying information pertaining to a subscription, extending the duration of a subscription and unsubscribing to Twinning notifications.

The notification URL passed by the client during the subscription step can be a Client-side Notification URL, or a Server-side Notification URL. Refer to [REST_NetAPI_NotificationChannel] for sequence flows illustrating the creation of a Notification Channel and obtaining a Server-side Notification URL on the server-side, and its use.

The resources:

– To subscribe to Twinning notifications, create a new resource under **http://{serverRoot}/twinning/{apiVersion}/{userId}/subscriptions**

– To retrieve list of active subscriptions, read the following resource **http://{serverRoot}/twinning/{apiVersion}/{userId}/subscriptions**

– To retrieve information about an individual subscription, read the following resource **http://{serverRoot}/twinning/{apiVersion}/{userId}/subscriptions/{subscriptionId}**

– To extend the duration of a subscription, update the following resource **http://{serverRoot}/twinning/{apiVersion}/{userId}/subscriptions/{subscriptionId}/duration**

– To cancel subscription to Twinning notifications delete the resource under **http://{serverRoot}/twinning/{apiVersion}/{userId}/subscriptions/{subscriptionId}**

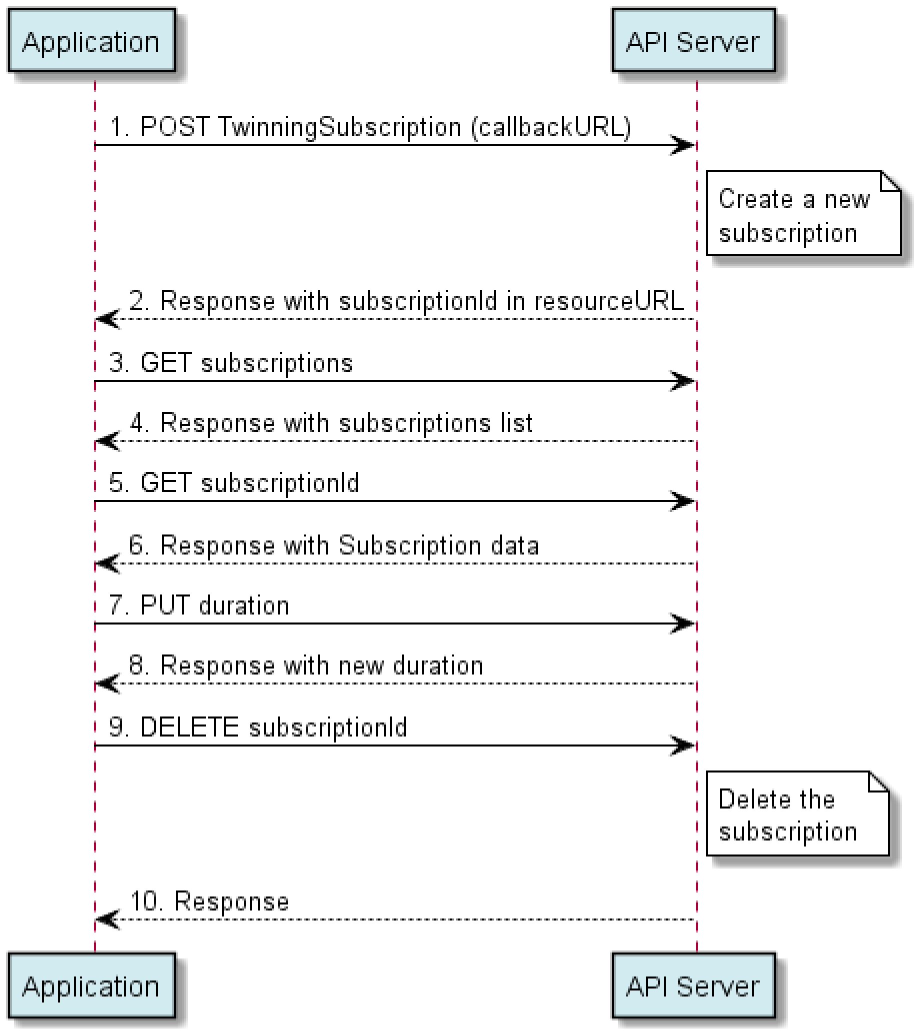
Application
API Server
1. POST TwinningSubscription (callbackURL)
Create a new subscription
2. Response with subscriptionId in resourceURL
3. GET subscriptions
4. Response with subscriptions list
5. GET subscriptionId
6. Response with Subscription data
7. PUT duration
8. Response with new duration
9. DELETE subscriptionId
Delete the subscription
10. Response
Application
API Server

Figure 2 Twinning notifications subscription

Outline of the flows:

1. An application subscribes to Twinning notifications using the POST method to submit the TwinningSubscription data structure to the resource containing all subscriptions.
2. The application receives the result resource URL containing the subscriptionId.
3. An application requests the list of active subscriptions, using the GET method to the resource containing all subscriptions.
4. The server returns subscriptions list belonging to the application in the response.
5. An application requests information pertaining to an individual subscription using the GET method to the resource.
6. The server returns subscription's data which includes duration, etc. in the response.
7. An application requests to extend the life of an individual subscription using the PUT method to the resource.
8. The server returns subscription's new duration in the response (note: duration returned in the response may be different from what was requested due to service provider's policy).
9. The application stops receiving notifications (on a given subscription) using DELETE with the resource URL containing the subscriptionId.
10. Deletion confirmation.

5.4.2 Activate a Twinning relationship (intra-MNO scenario)

The figure below shows a Twinning activation scenario in an intra-MNO scenario where both the Primary and the Secondary Devices are served by the same MNO.

The resources:

- **To** retrieve Twinning instructions from Primary or Secondary Device read the following resource including the "instruction" query parameter **http://{serverRoot}/twinning/{apiVersion}/{userId}/twinningRelations**
- To initiate a Twinning activation from Primary or Secondary Device, create a new resource under **http://{serverRoot}/twinning/{apiVersion}/{userId}/twinningRelations**

  Note: As depicted in figure below, Twinning activation SHOULD first be initiated from the Secondary Device as this results in an activation code which is subsequently used by the Primary Device in its Twinning activation request.

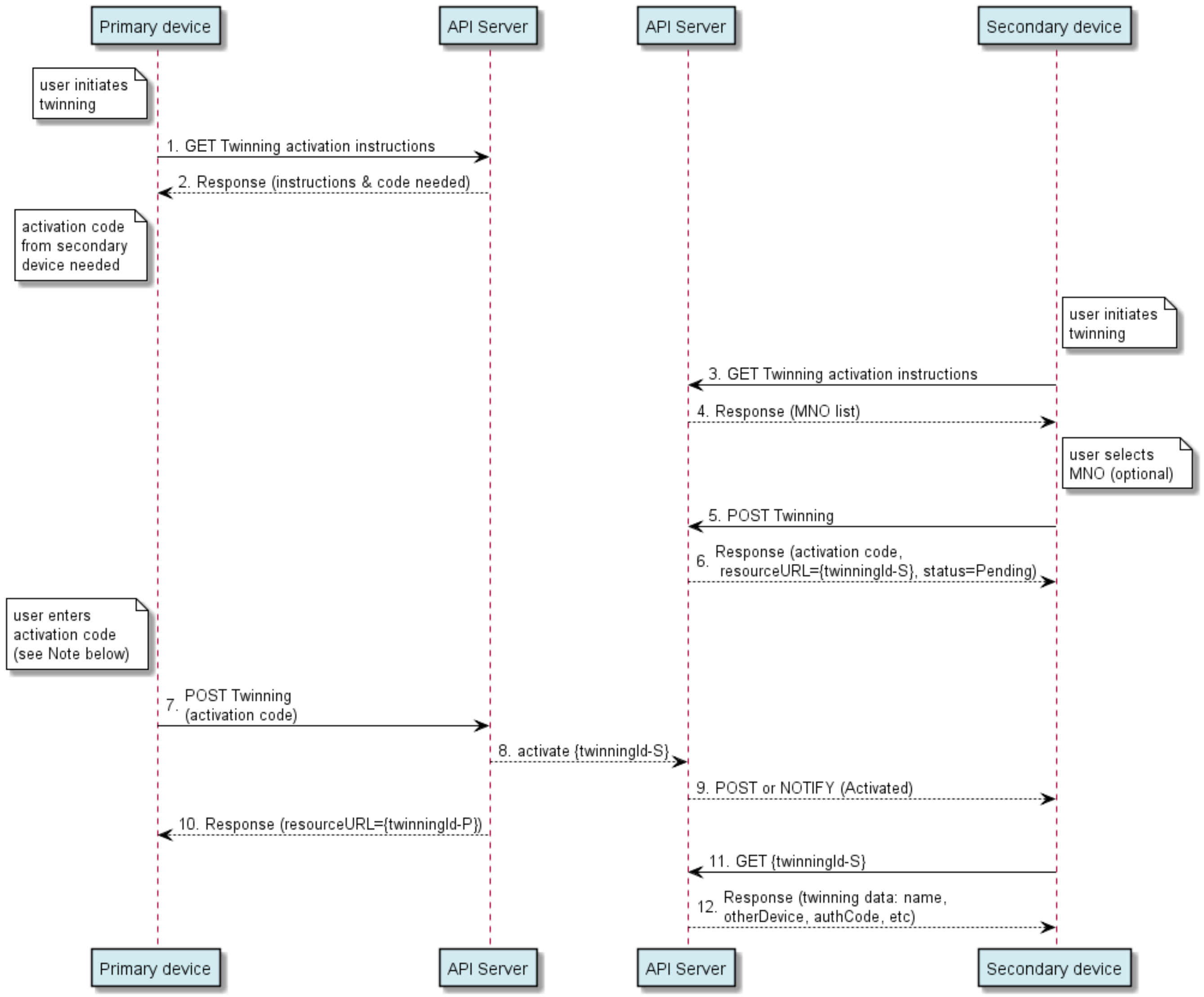
Primary device
API Server
API Server
Secondary device
user initiates twinning
1. GET Twinning activation instructions
2. Response (instructions & code needed)
activation code from secondary device needed
user initiates twinning
3. GET Twinning activation instructions
4. Response (MNO list)
user selects MNO (optional)
5. POST Twinning
6. Response (activation code, resourceURL={twinningId-S}, status=Pending)
user enters activation code (see Note below)
7. POST Twinning (activation code)
8. activate {twinningId-S}
9. POST or NOTIFY (Activated)
10. Response (resourceURL={twinningId-P})
11. GET {twinningId-S}
12. Response (twinning data: name, otherDevice, authCode, etc)
Primary device
API Server
API Server
Secondary device

Figure 3 Twinning activation (intra-MNO)

Outline of the flows:

1. Upon user’s demand, the application on the Primary Device initiates the Twinning activation by retrieving the twinning activation instruction from the server using the query parameter “instruction” = “ForPrimary”.
2. The server returns the Twinning activation instruction plus the need to enter a valid activation code
3. As per instruction user initiates the Twinning activation from the Secondary Device by retrieving the twinning activation instruction from the server using the query parameter “instruction =ForSecondary”.
4. The server returns the list of MNO’s from which the user selects the Primary Device’s MNO. Assuming the user doesn’t select any or selects the MNO which is the same as the one serves the Secondary Device.

   Note: step 4 can be ignored if inter-MNO Twinning activation is not supported.
5. Upon user’s confirmation, the application on the Secondary Device requests Twinning activation
6. The server returns an activation code and the resource URL for the newly created twinning resource which is in a “Pending” state.

   Note: The way the activation code is acquired from the Secondary and entered into the Primary Device is beyond the scope of this specification.
7. Upon entering the activation code into the Primary Device, the application requests Twinning activation.
8. The server creates a new twinning resource for the Primary Device, provisions the network accordingly and updates the twinning resource of the Secondary’s device (e.g. otherDevice parameters of Secondary’s {twinningId} resource are populated and its status is set to “On”) accordingly. At this point the twinning resource’s status for both the Primary and the Secondary is “On”
9. The Server notifies the Secondary Device of the twinning status change (from “Pending” to “On”). It is assumed that the Secondary earlier has subscribed to twinning events.
10. The Server confirms Twinning activation and returns in the body, the newly created twinning resource URL and other related information.
11. The application from the Secondary Device may optionally query Twinning data (e.g. in use cases where the Twinning of messages is also involved, the Secondary Device would need to have the Primary’s “authCode” in order to communicate with Primary’s network directly).
12. The server returns the entire Twinning data including the Primary Device’s “authCode” (see section 5.3.2.5 for details).

5.4.3 Retrieve Twinning relationship(s)

This figure below shows a scenario for an application reading the details of an individual Twinning relationship or retrieving the list of all the Twinning relationships a user (i.e. {userId}) is involved in.

The resources:

– To retrieve the existing Twinning relationship(s) a user/device is involved in, read the resource under **http://{serverRoot}/twinning/{apiVersion}/{userId}/twinningRelations**

– To retrieve information about an individual Twinning relationship, read the resource under **http://{serverRoot}/twinning/{apiVersion}/{userId}/twinningRelations/{twinningId}**

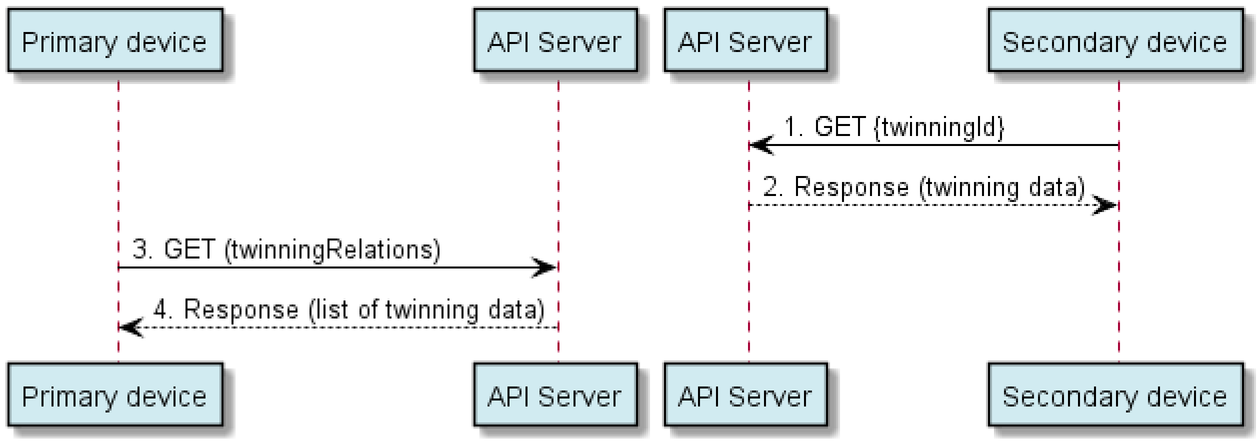
Primary device
API Server
API Server
Secondary device
1. GET {twinningId}
2. Response (twinning data)
3. GET (twinningRelations)
4. Response (list of twinning data)

Figure 4 Twinning data

Outline of the flows:

1. The application from the Secondary Device requests Twinning data of an individual twinning resource
2. The server returns the Twinning data
3. The application from the Primary Device requests to have the list of all the Twinning relationships it is involved in
4. The server returns the list of existing Twinning relationships including the details

5.4.4 Toggle On/Off a Twinning relationship

This figure below shows a scenario for an application reading the status of a Twinning relationship and toggling the Twinning status On/Off.

The resources:

– To retrieve the status of an individual Twinning relationship, read the resource under **http://{serverRoot}/twinning/{apiVersion}/{userId}/twinningRelations/{twinningId}/status**

– To set the status of a Twinning relationship to On or Off, update the following resource **http://{serverRoot}/twinning/{apiVersion}/{userId}/twinningRelations/{twinningId}/status**

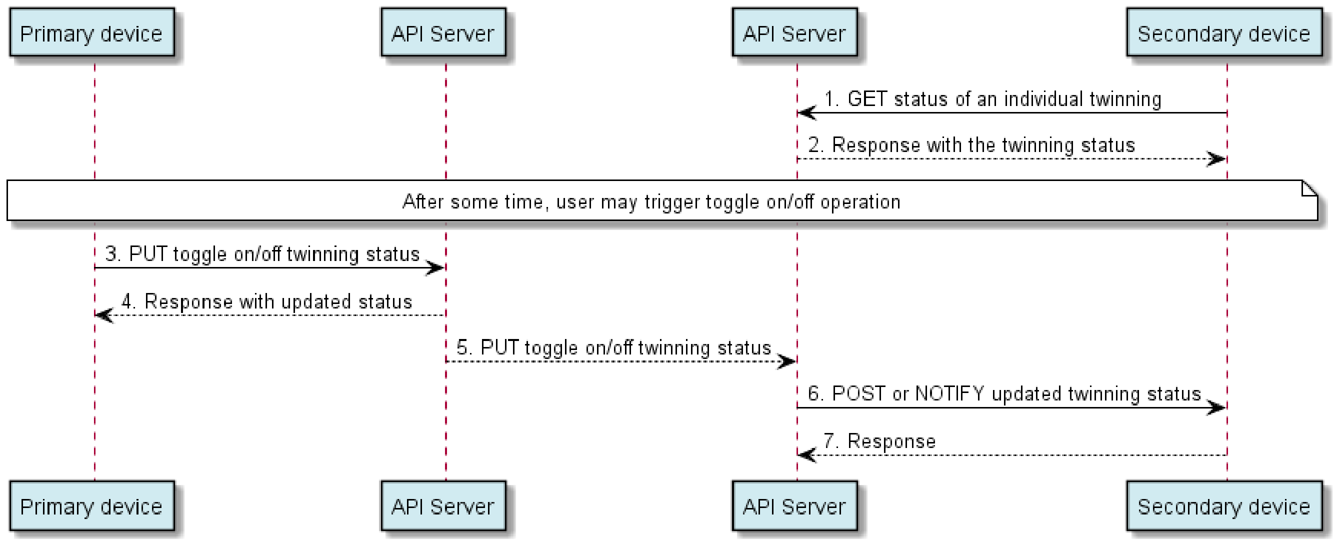
Primary device
API Server
API Server
Secondary device
1. GET status of an individual twinning
2. Response with the twinning status
After some time, user may trigger toggle on/off operation
3. PUT toggle on/off twinning status
4. Response with updated status
5. PUT toggle on/off twinning status
6. POST or NOTIFY updated twinning status
7. Response

Figure 5 Toggling Twinning status

Outline of the flows:

1. The application from the Secondary Device requests its Twinning status
2. The server returns the Twinning status as (e.g. status=On)
3. The application from the Primary Device (based on user's trigger) requests its Twinning status to be changed (e.g. toggled off)
4. The server confirms that the Twinning status is updated (e.g. status=Off)
5. The server updates the Secondary Device's resource status accordingly (e.g. status =Off)

   Note: in an inter-MNO scenario, this step is accomplished by one API Server invoking PUT operation onto the other API Server which is managing the associated resource (i.e./{twiningId}) for the Secondary Device. In intra-MNO scenario, the server may have other means of updating the required information however those means are out of scope of this specification.
6. The server notifies the Secondary Device that the Twinning status has been updated (e.g. toggled off). It is assumed that the Secondary Device has already subscribed toTwinning event notifications.
7. Notification Confirmation

5.4.5 Update Twinning relationships's user preference data

This figure below shows a scenario for an application on the Primary Device allowing the user to update the name of an existing Twinning relationship and also update Primary user's out-of-band notification settings.

The resources:

– To set the userPreference data of a Twinning relationship, update the following resource **http://{serverRoot}/twinning/{apiVersion}/{userId}/twinningRelations/{twinningId}/userPreference**

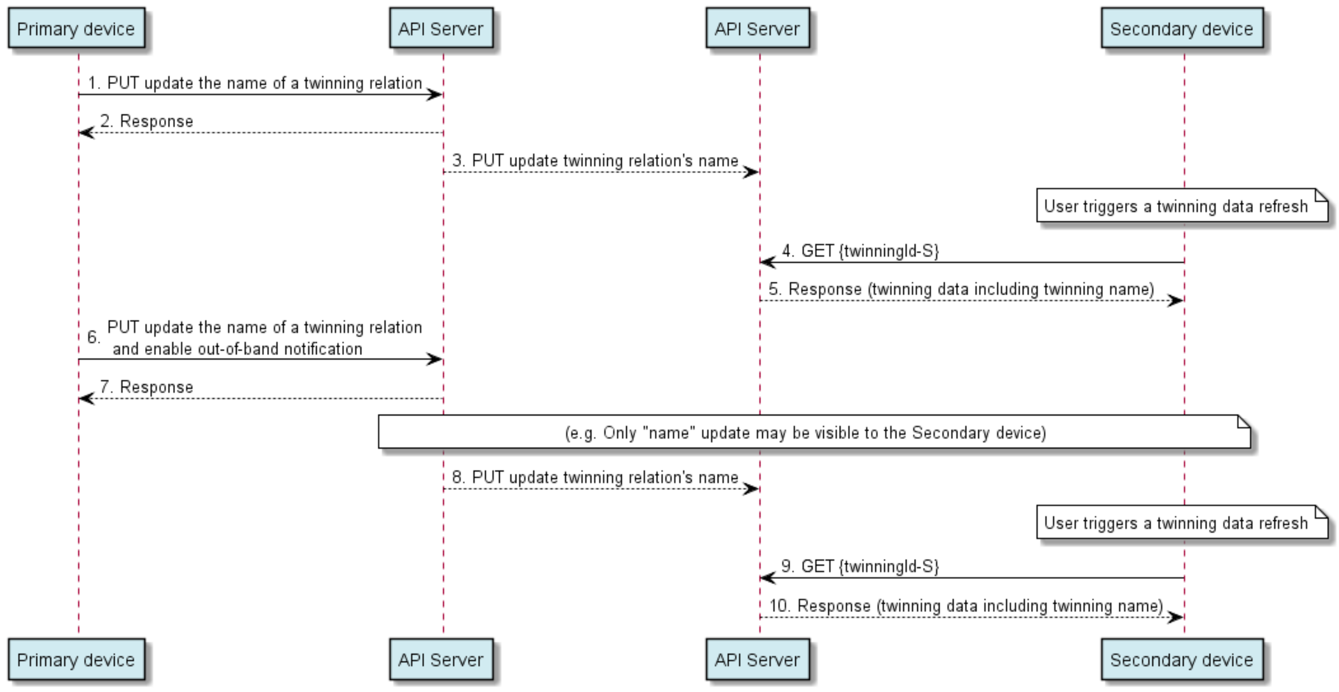
Primary device
API Server
API Server
Secondary device
1. PUT update the name of a twinning relation
2. Response
3. PUT update twinning relation's name
User triggers a twinning data refresh
4. GET {twinningId-S}
5. Response (twinning data including twinning name)
6. PUT update the name of a twinning relation and enable out-of-band notification
7. Response
(e.g. Only "name" update may be visible to the Secondary device)
8. PUT update twinning relation's name
User triggers a twinning data refresh
9. GET {twinningId-S}
10. Response (twinning data including twinning name)
Primary device
API Server
API Server
Secondary device

Figure 6 Update user-preference data

Outline of the flows:

1. The application from the Primary Device (based on user's wish) requests to update the name of an existing Twinning relationship (e.g. name=Watch)
2. The server confirms in the response the new userPreference setting
3. The server updates the Secondary device's resource with the new Twinning relationship name (if necessary)

   Note: in an inter-MNO scenario, this step is accomplished by one API Server invoking PUT operation onto the other API Server which is managing the associated resource (i.e./{twiningId}) for the Secondary device. In intra-MNO scenario, the server may have other means of updating the required information however those means are out of scope of this specification.
4. Upon user's trigger, the application on the Secondary device refreshes its Twinning data by requesting a GET {twinningId-S}
5. Server responds with the Twinning data which contains the new Twining relationship name.
6. The application from the Primary Device (based on user's wish) requests to update the name of the Twinning relationship while also enables the out-of-band Twinning notifications (e.g. by setting the outOfBandNotifyPrimaryUser flag to true and also providing a phone number at which he/she would like to receive the out-of-band notifications at (i.e. outOfBandNotifyPrimaryUserId is set to +19585550109))
7. The server confirms in the response the new userPreference setting
8. The server updates the Secondary Device's resource with the new Twinning relationship name (if necessary). Note, since out-of-band Twinning notifications setting is a meant to be Primary user's setting, such information MAY be hidden from the secondary

   Note: in an inter-MNO scenario, this step is accomplished by one API Server invoking PUT operation onto the other API Server which is managing the associated resource (i.e./{twiningId}) for the Secondary Device. In intra-MNO scenario, the server may have other means of updating the required information however those means are out of scope of this specification.
9. Upon user's trigger, the application on the Secondary Device refreshes its Twinning data by requesting a GET {twinningId-S}
10. Server responds with the Twinning data which contains the new Twining relationship name.

5.4.6 Deactivate a Twinning relationship

This figure below shows a scenario for an application permanently deactivating (i.e. removing) a Twinning relationship.

The resources:

– To permanently remove a Twinning relationship, delete the following resource **http://{serverRoot}/twinning/{apiVersion}/{userId}/twinningRelations/{twinningId}**

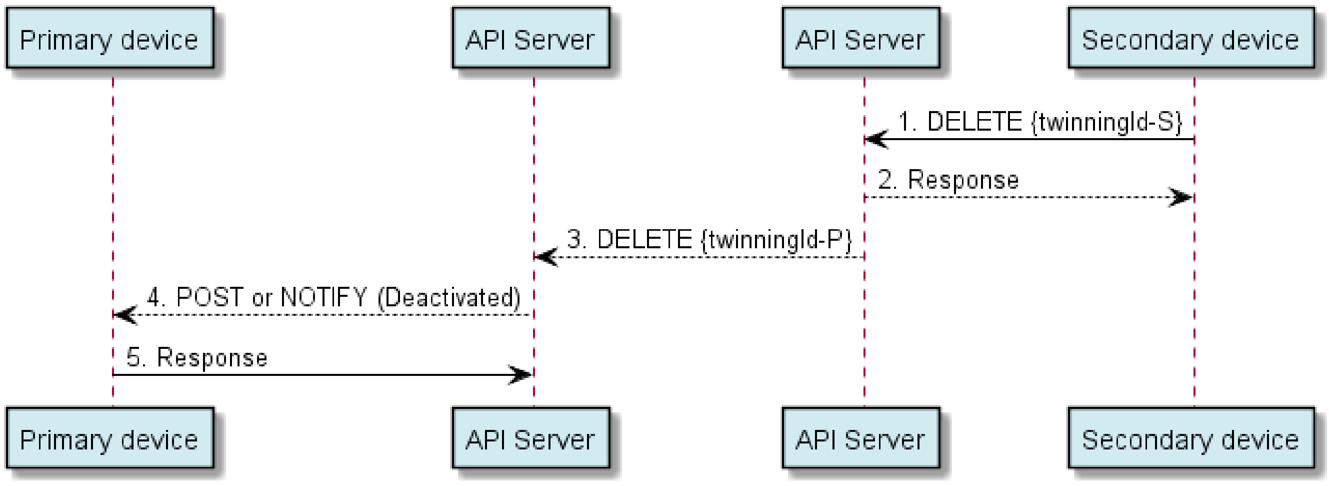

Figure 7 Twinning deactivation

Outline of the flows:

1. The application from the Secondary Device requests its Twinning relationship removed
2. The server confirms that the Twinning has been removed
3. The server removes the associated Primary Device's twinning resource

   Note: in an inter-MNO scenario, this step is accomplished by one API Server conveying DELETE operation onto the other API Server which is managing the associated resource (i.e./{twiningId}) for the Primary Device. In intra-MNO scenario, the server may have other means of removing the required information however those means are out of scope of this specification.
4. The server notifies the Primary Device of this event. Assuming that the Primary Device has already subscribed to Twining event notifications.
5. Notification Confirmation

5.4.7 Activate a Twinning relationship (inter-MNO scenario)

The figure below shows a Twinning activation scenario in an inter-MNO scenario where the Primary and the Secondary Devices are served by different MNO's.

The resources:

– **To** retrieve Twinning instructions from Primary or Secondary Device read the following resource including the "instruction" query parameter **http://{serverRoot}/twinning/{apiVersion}/{userId}/twinningRelations**

– To initiate a Twinning activation from Primary or Secondary Device, create a new resource under **http://{serverRoot}/twinning/{apiVersion}/{userId}/twinningRelations**

– To retrieve an inter-MNO activation code use resource under. **http://{serverRoot}/twinning/{apiVersion}/interMnoActivationCode**

– To set Secondary's Twinning resource data, update the resource under **http://{serverRoot}/twinning/{apiVersion}/{userId}/twinningRelations/{twinningId}**

  Note: As depicted in the figure below, Twinning activation SHOULD first be initiated from the Secondary Device as this results in an activation code which is subsequently used by the Primary Device in its Twinning activation request.

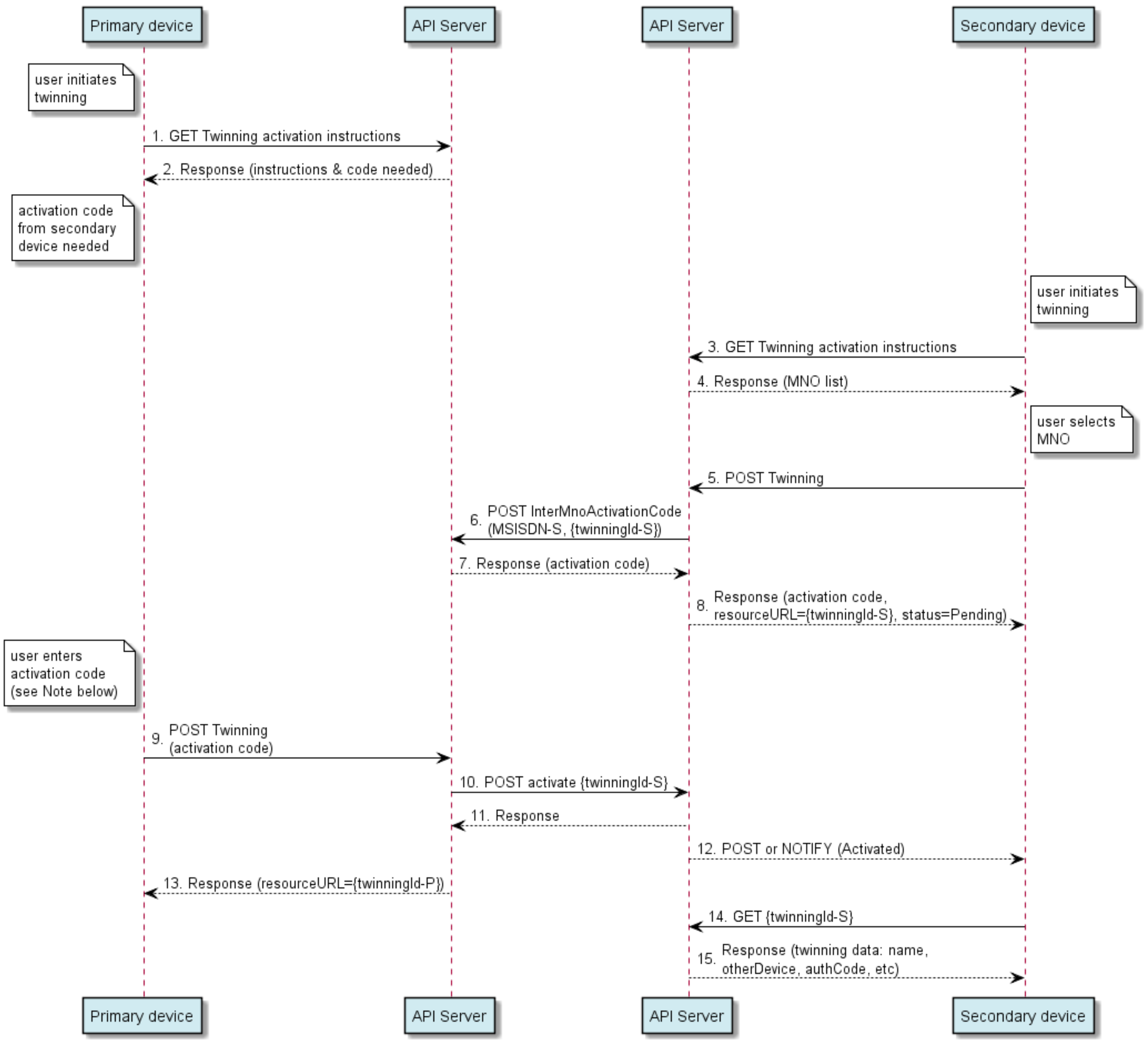
Primary device
API Server
API Server
Secondary device
user initiates twinning
1. GET Twinning activation instructions
2. Response (instructions & code needed)
activation code from secondary device needed
user initiates twinning
3. GET Twinning activation instructions
4. Response (MNO list)
user selects MNO
5. POST Twinning
6. POST InterMnoActivationCode (MSISDN-S, {twinningId-S})
7. Response (activation code)
8. Response (activation code, resourceURL={twinningId-S}, status=Pending)
user enters activation code (see Note below)
9. POST Twinning (activation code)
10. POST activate {twinningId-S}
11. Response
12. POST or NOTIFY (Activated)
13. Response (resourceURL={twinningId-P})
14. GET {twinningId-S}
15. Response (twinning data: name, otherDevice, authCode, etc)

Figure 8 Twinning activation (inter-MNO)

Outline of the flows:

1. Upon user’s demand, the application on the Primary Device initiates the Twinning activation by retrieving the twinning activation instruction from the server using the query parameter “instruction =ForPrimary”.
2. The server returns the Twinning activation instruction plus the need to enter a valid activation code
3. As per instruction (seen on the Primary device) user initiates the Twinning activation from the Secondary Device by retrieving the twinning activation instruction from the server using the query parameter “instruction =ForSecondary”.
4. The server returns the list of MNO’s from which the user selects the Primary Device’s MNO. Assuming the user selects an MNO which is different from the one serves the Secondary’s device.
5. Upon user’s confirmation, the application on the Secondary Device requests Twinning activation
6. The server realizes that that the Primary Device belongs to a different MNO. Hence, after creation of the Twinning resource for the Secondary (i.e. /{twinningId}), the server invokes a request onto the API Server of the Primary Device

for an interMNO activation code. The inter-MNO activationCode request contains the required Secondary Device's information (e.g. MSISDN and /{twinningId}).

7. The Primary Device's API Server returns an activation code.
8. The Secondary Device's Server returns the activation code and the resource URL (i.e. /{twinningId}) of the newly created resource which is in a "Pending" state.
    - Note: The way the activation code is acquired from the Secondary and entered into the Primary Device is beyond the scope of this specification.
9. Upon entering the activation code into the Primary Device, the application requests Twinning activation.
10. The server creates a new twinning resource for the Primary Device, provisions the network on the Primary's side and invokes a POST request onto the API Server of the Secondary's device in order to update the twinning resource of the Secondary's device (e.g. otherDevice parameters of Secondary's /{twinningId} resource are populated and effectively its status is set to "On"). At this point the twinning resource's status for both the Primary and the Secondary is "On".

    Note: This step is accomplished by one API Server invoking POST operation onto the other API Server which is managing the associated resource (i.e./{twiningId}) for the Secondary Device. The Primary's API Server receives the resourceURL (i.e. / {twiningId}) of the Secondary as part of step 6.
11. The Secondary's API server confirms a successful operation.
12. The Server notifies the Secondary Device of the twinning status change (from "Pending" to "On"). It is assumed that the Secondary earlier has subscribed to twinning events.
13. The Server confirms Twinning activation and returns in the body, the newly created Primary's twinning resource URL and other related information.
14. The application from the Secondary Device may optionally request Twinning data (e.g. in cases where the Twinning of messages is also involved, the Secondary Device needs to know of the Primary's "authCode" in order to communicate with Primary's network directly).
15. The server returns the entire Twinning data including the Primary Device's "authCode" (see section 5.3.2.5 for details).

6. Detailed specification of the resources

The following applies to all resources defined in this specification regardless of the representation format (i.e. XML, JSON):

- Reserved characters in URL variables (parts of a URL denoted below by a name in curly brackets) MUST be percent-encoded according to [RFC3986]. Note that this always applies, no matter whether the URL is used as a Request URL or inside the representation of a resource (such as in "resourceURL" and "link" elements).
- If a user identifier (e.g. address, participantAddress, etc.) of type anyURI is in the form of an MSISDN, it MUST be defined as a global number according to [RFC3966] (e.g. tel:+19585550100). The use of characters other than digits and the leading "+" sign SHOULD be avoided in order to ensure uniqueness of the resource URL. This applies regardless of whether the user identifier appears in a URL variable or in a parameter in the body of an HTTP message.
- If an equipment identifier of type anyURI is in the form of a SIP URI, it MUST be defined according to [RFC3261].
- If a user identifier (e.g. address, userId, etc) of type anyURI is in the form of an Anonymous Customer Reference (ACR), it MUST be defined according to Appendix H of [REST_NetAPI_ACR].
    - o The ACR 'auth' is a supported reserved keyword, and MUST NOT be assigned as an ACR to any particular end user. See G.1.2 for details regarding the use of this reserved keyword.
- For requests and responses that have a body, the following applies: in the requests received, the server SHALL support JSON and XML encoding of the parameters in the body. The Server SHALL return either JSON or XML encoded parameters in the response body, according to the result of the content type negotiation as specified in [REST_NetAPI_Common]. In notifications to the Client, the server SHALL use either XML or JSON encoding, depending on which format the client has specified in the related subscription. The generation and handling of the JSON representations SHALL follow the rules for JSON encoding in HTTP Requests/Responses as specified in [REST_NetAPI_Common].

6.1 Resource: Twinning relationships

The resource used is:

//{serverRoot}/twinning/{apiVersion}/{userId}/twinningRelations

This resource is used for creating a new twinning relationship as well as retrieving the list of existing twinning relationships for a specified user/device. This resource is also used to retrieve Twinning activation instructions from the network.

6.1.1 Request URL variables

The following request URL variables are common for all HTTP methods:

Name	Description
serverRoot	Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI
apiVersion	Version of the API client wants to use. The value of this variable is defined in section 5.2
userId	Identifier of the user on whose behalf the application acts Examples: tel:+19585550100, acr:pseudonym123, sip:alice@example.com

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.1.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to RESTful Network API for Twinning Devices, see section 7.

6.1.3 GET

This operation is used for retrieving the existing Twinning relationship(s) a user (i.e. {userId}) is involved in.

Supported parameters in the query string of the Request URL are:

Name	Type/Values	Optional	Description
instruction	xsd:string	Yes	Controls whether Twinning instruction is required to be returned in the body of the GET response. Supported values for “instruction” query string SHALL be “ForPrimary” and “ForSecondary”. • If “instruction” query parameter is absent, GET response body SHALL NOT include the “instruction” element. • If instruction=ForPrimary, GET response body SHALL include the “instruction” element. The instruction returned by the server is tailored for usage in the context of a Primary Device (i.e. the request is invoked by a Primary Device). The instruction MAY exclude the MNO list. • If instruction=ForSecondary, GET response body SHALL include the “instruction” element. The instruction returned by the server is tailored for usage in the context of a Secondary Device (i.e. the request is invoked by a Secondary Device). The instruction SHALL include the MNO list.

6.1.3.1 Example 1: Retrieve a Primary Device’s existing Twinning relationships (Informative)

In this example the query is invoked by the user’s Primary Device (i.e. {userId}) which is twinned with two Secondary Devices (a wearable watch and a connected car). Currently, the wearable watch’s twinning relationship is toggled off.

6.1.3.1.1 Request

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations  HTTP/1.1
Host: example.com
Accept: application/xml
```

6.1.3.1.2 Response

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 02:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinningList xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <twinning>
    <role>Primary</role>
    <name>MyCar</name>
    <otherDevice>
```

```
      <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550101/twinningRelations/1900</resourceURL>
      <address>tel:+19585550101</address>
      <mno>
        <name>MNO-A</name>
        <ncc>310410</ncc>
      </mno>
    </otherDevice>
    <twinningStatus>On</twinningStatus>
    <ownAddress>tel:+19585550100</ownAddress>
    <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/1800</resourceURL>
 </twinning>
 <twinning>
   <role>Primary</role>
   <name>MyWatch</name>
   <otherDevice>
     <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550102/twinningRelations/222</resourceURL>
     <address>tel:+19585550102</address>
     <mno>
       <name>MNO-A</name>
       <ncc>310410</ncc>
     </mno>
   </otherDevice>
   <twinningStatus>Off</twinningStatus>
   <ownAddress>tel:+19585550100</ownAddress>
   <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/111</resourceURL>
  </twinning>
  <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations</resourceURL>
</twinning:twinningList>
```

6.1.3.2 Example 2: Retrieve a Secondary Device's existing Twinning relationship using "auth" keyword (Informative)

In this example, the query is invoked by the user's Secondary Device (i.e. {userId}) which may be twinned with only one Primary Device at a time. In this example the user/device is identified using the OAuth access token contained in the Authorization header.

6.1.3.2.1 Request

```
GET /exampleAPI/twinning/v1/acr%3Aauth/twinningRelations  HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d77-4aa6-a404-2bc19b5cf333
Accept: application/xml
```

6.1.3.2.2 Response

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 02:54:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinningList xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <twinning>
    <role>Secondary</role>
    <name>MyCar</name>
    <otherDevice>
      <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/1900</resourceURL>
      <address>tel:+19585550100</address>
      <mno>
        <name>MNO-A</name>
       <ncc>310410</ncc>
     </mno>
   </otherDevice>
    <twinningStatus>On</twinningStatus>
    <ownAddress>tel:+19585550101</ownAddress>
    <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550101/twinningRelations/1800</resourceURL>
  </twinning>
  <resourceURL>http://exampleAPI/twinning/v1/acr%3Aauth/twinningRelations</resourceURL>
</twinning:twinningList>
```

6.1.3.3 Example 3: Retrieve Twinning instruction from a Secondary Device (Informative)

6.1.3.3.1 Request

```
GET /exampleAPI/twinning/v1/acr%3Aauth/twinningRelations?instruction=ForSecondary  HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d77-4aa6-a404-2bc19b5cf333
Accept: application/xml
```

6.1.3.3.2 Response

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 02:54:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinningList xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <instruction>
    <mno>
      <name>MNO-A</name>
      <ncc>310410</ncc>
    </mno>
    <mno>
      <name>MNO-B</name>
      <ncc>23415</ncc>
    </mno>
```

```
      <mno>
         <name>MNO-C</name>
          <ncc>310260</ncc>
      </mno>
      <mno>
         <name>MNO-D Wireless</name>
         <ncc>23208</ncc>
      </mno>
      <activationInstruction>If the Primary Device with which this device is intended to be twinned is served by a different mobile service
provider please select the Primary Device's mobile service provider from the list</activationInstruction>
   </instruction>
   <resourceURL>http://exampleAPI/twinning/v1/acr%3Aauth/twinningRelations</resourceURL>
</twinning:twinningList>
```

6.1.3.4 Example 4: Retrieve Twinning instruction from a Primary Device (Informative)

In this example, the Primary Device is already in two Twinning relationships and intends to form a third Twinning relationship.

6.1.3.4.1 Request

```
GET /exampleAPI/twinning/v1/acr%3Aauth/twinningRelations?instruction=ForPrimary  HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d77-4aa6-a404-2bc19b511111
Accept: application/xml
```

6.1.3.4.2 Response

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 02:54:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinningList xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
   <twinning>
      <role>Primary</role>
      <name>MyCar</name>
      <otherDevice>
         <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550101/twinningRelations/1900</resourceURL>
         <address>tel:+19585550101</address>
         <mno>
            <name>MNO-A</name>
           <ncc>310410</ncc>
        </mno>
      </otherDevice>
      <twinningStatus>On</twinningStatus>
      <ownAddress>tel:+19585550100</ownAddress>
      <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/1800</resourceURL>
  </twinning>
  <twinning>
     <role>Primary</role>
     <name>MyWatch</name>
     <otherDevice>
```

```
        <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550102/twinningRelations/222</resourceURL>
        <address>tel:+19585550102</address>
        <mno>
          <name>MNO-A</name>
          <ncc>310410</ncc>
        </mno>
      </otherDevice>
      <twinningStatus>Off</twinningStatus>
      <ownAddress>tel:+19585550100</ownAddress>
      <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/111</resourceURL>
    </twinning>
    <instruction>
       <activationInstruction>please enter an activation code acquired from the Secondary Device with which this device is intended to be
twinned with</activationInstruction>
    </instruction>
    <resourceURL>http://exampleAPI/twinning/v1/acr%3Aauth/twinningRelations</resourceURL>
</twinning:twinningList>
```

6.1.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET, POST' field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

6.1.5 POST

This operation is used to initiate and create a new Twinning relationship.

In order to create a Twinning relationship, both the Secondary and the Primary Device independently need to invoke the POST method.

Invocation of the POST method by the Secondary Device SHALL result in creation of a new twinning resource (by the server) representing the Secondary Device's side of the twinning relationship in a "Pending" status.

Subsequent invocation of the POST method by the Primary Device SHALL result in creation of a new twinning resource (by the server) representing the Primary Device's side of the twinning relationship. A successful POST operation from the Primary Device SHALL result in both twinning resources linked together in an "On" status.

6.1.5.1 Example 1: A Secondary Device initiates a new Twinning relationship (Informative)

In this example, the Secondary Device is being twinned with a Primary Device which is served by the same MNO, hence the optional "primaryMno" element is not included in the POST request body. See example 2 below which is the continuation of example 1.

The response contains the resourceURL of the twinning resource created. When the twinning resource changes from its initial state of "pending" to "On", the application on the Secondary Device SHALL receive an appropriate notification. Optionally, if the application doesn't subscribe to twinning events, it may periodically query the status of the twinning resource using the resourceURL received in the response.

6.1.5.1.1 Request

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550103/twinningRelations HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Secondary</role>
</twinning:twinning>
```

6.1.5.1.2 Response

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550103/twinningRelations/333
Content-Type: application/xml
Content-Length: nnnn
Date: Tue, 03 Feb 2015 21:32:52 GMT

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Secondary</role>
  <activationCode>
    <code>711711</code>
    <mno>
      <name>MNO-A</name>
      <ncc>310410</ncc>
    </mno>
  </activationCode>
  <twinningStatus>Pending</twinningStatus>
  <ownAddress>tel:+19585550103</ownAddress>
  <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550103/twinningRelations/333</resourceURL>
</twinning:twinning>
```

6.1.5.2 Example 2: A Primary Device initiates and activates a new Twinning relationship (Informative)

6.1.5.2.1 Request

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Primary</role>
  <name>FamilySUV</name>
  <activationCode>
    <code>711711</code>
  </activationCode>
</twinning:twinning>
```

6.1.5.2.2 Response

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/2000
Content-Type: application/xml
Content-Length: nnnn
Date: Tue, 03 Feb 2015 21:32:52 GMT

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Primary</role>
  <name>FamilySUV</name>
  <otherDevice>
     <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550103/twinningRelations/333</resourceURL>
     <address>tel:+19585550103</address>
     <mno>
        <name>MNO-A</name>
        <ncc>310410</ncc>
     </mno>
  </otherDevice>
  <activationCode>
     <code>711711</code>
     <mno>
        <name>MNO-A</name>
        <ncc>310410</ncc>
     </mno>
  </activationCode>
  <twinningStatus>On</twinningStatus>
  <ownAddress>tel:+19585550100</ownAddress>
  <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/2000</resourceURL>
</twinning:twinning>
```

6.1.5.3 Example 3: A Primary Device initiates and activates a new Twinning relationship, response with HTTP 202 Accepted (Informative)

In this example, the server returns an HTTP 202 Accepted response with the body including “twinningStatus” = “Pending” to mean that, this is not the final status of the Twinning resource.

The application client can check the twinning status using a GET operation on the provided resourceURL (of the created resource provided in the response body or the Location header) or receive a notification of the final status of the twinning resource once the execution of the request is complete (see section 6.8.5.1 for an “Activated” notification example).

6.1.5.3.1 Request

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Primary</role>
  <name>FamilySUV</name>
  <activationCode>
    <code>711711</code>
  </activationCode>
</twinning:twinning>
```

6.1.5.3.2 Response

```
HTTP/1.1 202 Accepted
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/2000
Content-Type: application/xml
Content-Length: nnnn
Date: Tue, 03 Feb 2015 21:32:52 GMT

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Primary</role>
  <name>FamilySUV</name>
  <otherDevice>
    <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550103/twinningRelations/333</resourceURL>
    <address>tel:+19585550103</address>
    <mno>
      <name>MNO-A</name>
      <ncc>310410</ncc>
    </mno>
  </otherDevice>
  <activationCode>
    <code>711711</code>
    <mno>
      <name>MNO-A</name>
      <ncc>310410</ncc>
    </mno>
  </activationCode>
  <twinningStatus>Pending</twinningStatus>
  <ownAddress>tel:+19585550100</ownAddress>
  <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/2000</resourceURL>
</twinning:twinning>
```

6.1.5.4 Example 4: A Primary Device initiates and activates a new Twinning relationship, response with location of created resource (Informative)

6.1.5.4.1 Request

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Primary</role>
  <name>FamilySUV</name>
  <activationCode>
    <code>711711</code>
  </activationCode>
</twinning:twinning>
```

6.1.5.4.2 Response

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/2000
Content-Type: application/xml
Content-Length: nnnn
Date: Tue, 03 Feb 2015 21:32:52 GMT

<?xml version="1.0" encoding="UTF-8"?>
<common:resourceReference xmlns:common="urn:oma:xml:rest:netapi:common:1">
   <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/2000</resourceURL>
 </common:resourceReference>
```

6.1.5.5 Example 5: A Secondary Device initiates a new inter-MNO Twinning relationship (Informative)

In this example, the Secondary Device (a Connected Car) is being twinned with a Primary Device which is served by another MNO, as identified by “primaryMno” element in the POST request body. See example 6 below which is the continuation of this example. Also see examples in sections 6.2.3.2, 6.2.4.1 and 6.6.5.1.

6.1.5.5.1 Request

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Secondary</role>
  <primaryMno>
    <name>MNO-A</name>
    <ncc>310410</ncc>
  </primaryMno>
```

```
</twinning:twinning>
```

6.1.5.5.2 Response

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789
Content-Type: application/xml
Content-Length: nnnn
Date: Tue, 03 Feb 2015 21:32:52 GMT

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Secondary</role>
  <activationCode>
    <code>123123</code>
    <mno>
      <name>MNO-A</name>
      <ncc>310410</ncc>
    </mno>
  </activationCode>
  <twinningStatus>Pending</twinningStatus>
  <ownAddress>tel:+19585550104</ownAddress>
  <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789</resourceURL>
</twinning:twinning>
```

6.1.5.6 Example 6: A Primary Device initiates and activates a new inter-MNO Twinning relationship (Informative)

6.1.5.6.1 Request

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Primary</role>
  <name>ConnectedCar</name>
  <activationCode>
    <code>123123</code>
  </activationCode>
</twinning:twinning>
```

6.1.5.6.2 Response

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/100
Content-Type: application/xml
Content-Length: nnnn
Date: Tue, 03 Feb 2015 21:32:52 GMT

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Primary</role>
  <name>ConnectedCar</name>
  <otherDevice>
    <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789</resourceURL>
    <address>tel:+19585550104</address>
    <mno>
      <name>MNO-B</name>
      <ncc>310260</ncc>
    </mno>
  </otherDevice>
  <activationCode>
    <code>123123</code>
    <mno>
      <name>MNO-A</name>
      <ncc>310410</ncc>
    </mno>
  </activationCode>
  <twinningStatus>On</twinningStatus>
  <ownAddress>tel:+19585550100</ownAddress>
  <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/100</resourceURL>
</twinning:twinning>
```

6.1.5.7 Example 7: A Primary Device initiates Twinning relationship activation, failure due to an expired activation code (Informative)

6.1.5.7.1 Request

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Primary</role>
  <name>FamilySUV</name>
  <activationCode>
    <code>711711</code>
  </activationCode>
</twinning:twinning>
```

6.1.5.7.2 Response

```
HTTP/1.1 400 Bad Request
Date: Tue, 03 Feb 2015 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <serviceException>
      <messageId>SVC2004</messageId>
      <text>Invalid input value for %1 %2: %3</text>
      <variables>activationCode</variables>
      <variables>711711</variables>
      <variables>has expired</variables>
  </serviceException>
</common:requestError>
```

6.1.5.8 Example 8: A Secondary Device initiates Twinning relationship activation, failure due to undefined twinning role (Informative)

The twinning role must either be provided in the request or somehow known to the server in advance (e.g. in advance pre-configuration information at the server). In this example, the client does not provide the role parameter in the request and the server does not have a way to determine the role. This results into an error as shown below.

6.1.5.8.1 Request

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1"/>
```

6.1.5.8.2 Response

```
HTTP/1.1 400 Bad Request
Date: Tue, 03 Feb 2015 12:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <serviceException>
      <messageId>SVC1014</messageId>
      <text>Role is required to form a Twinning relationship</text>
  </serviceException>
</common:requestError>
```

6.1.5.9 Example 9: A Primary device initiates Twinning relationship activation, failure due to a policy error such as “maximum number of twinning relationships reached” (Informative)

This example demonstrates, Twinning relationship activation failure due to the Service Provider’s policy such as “maximum number of Twinning relationships reached”.

Similarly, other service provider’s policy exceptions (e.g. Twinning feature not available to prepaid account types) may use POL2000 error code with appropriate values for the two variables as demonstrated in the following example.

6.1.5.9.1 Request

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Primary</role>
  <name>sportsBand</name>
  <activationCode>
     <code>713713</code>
  </activationCode>
</twinning:twinning>
```

6.1.5.9.2 Response

```
HTTP/1.1 403 Forbidden
Date: Thu, 20 Nov 2014 21:51:51 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
   <policyException>
    <messageId>POL2000</messageId>
    <text>The following policy error occurred: %1. Error code is %2</text>
    <variables>Maximum number of Twinning relationships reached. Delete an existing Twinning relationship and try again</variables>
    <variables>E234</variables>
  </policyException>
</common:requestError>
```

6.1.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, POST’ field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

6.2 Resource: An individual Twinning relationship

The resource used is:

//{serverRoot}/twinning/{apiVersion}/{userId}/twinningRelations/{twinningId}

This resource is used to manage (read, update and delete) an individual Twinning relationship.

6.2.1 Request URL variables

The following request URL variables are common for all HTTP methods:

Name	Description
serverRoot	Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI
apiVersion	Version of the API client wants to use. The value of this variable is defined in section 5.2
userId	Identifier of the user on whose behalf the application acts Examples: tel:+19585550100, acr:pseudonym123, sip:alice@example.com
twinningId	Identifier of the twinning relationship generated by the server during creation of a Twinning relationship

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.2.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to RESTful Network API for Twinning Devices, see section 7.

6.2.3 GET

This operation is used for retrieving the information about an individual Twinning relationship.

6.2.3.1 Example 1: Retrieve an individual Twinning relationship from Primary Device (Informative)

6.2.3.1.1 Request

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/100  HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d77-4aa6-a404-2bc19b5cf999
Accept: application/xml
```

6.2.3.1.2 Response

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 04:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Primary</role>
  <name>ConnectedCar</name>
  <otherDevice>
    <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789</resourceURL>
    <address>tel:+19585550104</address>
    <mno>
      <name>MNO-B</name>
      <ncc>310260</ncc>
    </mno>
  </otherDevice>
  <activationCode>
    <code>123123</code>
    <mno>
      <name>MNO-A</name>
      <ncc>310410</ncc>
    </mno>
  </activationCode>
  <twinningStatus>On</twinningStatus>
  <ownAddress>tel:+19585550100</ownAddress>
  <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/100</resourceURL>
</twinning:twinning>
```

6.2.3.2 Example 2: Retrieve an individual Twinning relationship from Secondary Device (Informative)

6.2.3.2.1 Request

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789  HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d77-4aa6-a404-2bc19b5cf
Accept: application/xml
```

6.2.3.2.2 Response

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 04:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Secondary</role>
  <activationCode>
    <code>123123</code>
    <mno>
      <name>MNO-A</name>
      <ncc>310410</ncc>
    </mno>
  </activationCode>
  <twinningStatus>Pending</twinningStatus>
  <ownAddress>tel:+19585550104</ownAddress>
  <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789</resourceURL>
</twinning:twinning>
```

6.2.3.3 Example 3: Retrieve information about a non-existent individual Twinning relationship (Informative)

6.2.3.3.1 Request

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/399621  HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d77-4aa6-a404-2bc19b5cf
Accept: application/xml
```

6.2.3.3.2 Response

```
HTTP/1.1 404 Not Found
Content-Type: application/xml
Content-Length: nnnn
Date: Fri, 11 Apr 2014 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <serviceException>
    <messageId>SVC0004</messageId>
    <text>No valid addresses provided in message part %1</text>
    <variables>Request-URI</variables>
  </serviceException>
</common:requestError>
```

6.2.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, POST, DELETE’ field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

6.2.5 POST

In inter-MNO Twinning activation, this operation is used (in between the MNO servers) to update and turn “On” (i.e. activate) a “Pending” Twinning relationship.

In inter-MNO scenarios, once the Primary Device side of the twinning relationship is activated, the Primary Device’s server SHALL invoke the POST method onto the Secondary Device’s server in order to update the twinning information on the Secondary’s side and effectively change its twinning status to “On” (the default status resulting from this operation).

Note: In intra-MNO twinning relationship since the same server has access to the twinning resources, the server normally relies on internal means to make the update as opposed to using the POST method.

6.2.5.1 Example 1: Activate an inter-MNO “Pending” Twinning relationship (Informative)

6.2.5.1.1 Request

```
POST  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789 HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:interMnoTwinningActivation xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <name>ConnectedCar</name>
  <otherDevice>
    <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/100</resourceURL>
    <address>tel:+19585550100</address>
    <mno>
      <name>MNO-A</name>
      <ncc>310410</ncc>
    </mno>
  </otherDevice>
  <authCode>6d0d-4aa6-a404</authCode>
  <primaryDeviceVMAddress>tel:+19585550199</primaryDeviceVMAddress>
</twinning:interMnoTwinningActivation>
```

6.2.5.1.2 Response

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 04:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinning xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <role>Secondary</role>
  <name>ConnectedCar</name>
  <otherDevice>
    <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/100</resourceURL>
    <address>tel:+19585550100</address>
    <mno>
      <name>MNO-A</name>
      <ncc>310410</ncc>
    </mno>
  </otherDevice>
  <activationCode>
    <code>123123</code>
    <mno>
      <name>MNO-A</name>
      <ncc>310410</ncc>
    </mno>
  </activationCode>
  <twinningStatus>On</twinningStatus>
  <authCode>6d0d-4aa6-a404</authCode>
  <primaryDeviceVMAddress>tel:+19585550199</primaryDeviceVMAddress>
  <ownAddress>tel:+19585550104</ownAddress>
  <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789</resourceURL>
</twinning:twinning>
```

6.2.5.2 Example 2: Updating information about an already active Twinning relationship fails (Informative)

In this example the client attempts to change otherDevice's information of an already active Twinning relationship. This operation is not allowed as indicated in the response.

6.2.5.2.1 Request

```
POST  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789 HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:interMnoTwinningActivation xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <name>ConnectedCar</name>
  <otherDevice>
    <resourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550101/twinningRelations/101</resourceURL>
    <address>tel:+19585550101</address>
    <mno>
      <name>MNO-A</name>
      <ncc>310410</ncc>
    </mno>
  </otherDevice>
  <authCode>6d0d-4aa6-a404</authCode>
</twinning:interMnoTwinningActivation>
```

6.2.5.2.2 Response

```
HTTP/1.1 403 Forbidden
Date: Tue, 03 Feb 2015 04:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
   <policyException>
    <messageId>POL1038</messageId>
    <text>Updating Twinning data of an active Twinning relationship is not allowed</text>
  </policyException>
</common:requestError>
```

6.2.6 DELETE

This operation is used to delete a Twinning relationship.

In inter-MNO scenarios one server SHALL invoke the DELETE method onto the other device’s server in order to delete the corresponding twinning resource accordingly

The server responds to a DELETE request with an HTTP 204 No Content response.

6.2.6.1 Example: Delete a Twinning relationship (Informative)

6.2.6.1.1 Request

```
DELETE /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
```

6.2.6.1.2 Response

```
HTTP/1.1 204 No Content
Date: Tue, 03 Feb 2015 06:55:59 GMT
```

6.3 Resource: Twinning status

The resource used is:

//{serverRoot}/twinning/{apiVersion}/{userId}/twinningRelations/{twinningId}/status

This resource is used to manage (read, update) the status of a Twinning relationship.

6.3.1 Request URL variables

The following request URL variables are common for all HTTP methods:

Name	Description
serverRoot	Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI
apiVersion	Version of the API client wants to use. The value of this variable is defined in section 5.2
userId	Identifier of the user on whose behalf the application acts Examples: tel:+19585550100, acr:pseudonym123, sip:alice@example.com
twinningId	Identifier of the twinning

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.3.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to RESTful Network API for Twinning Devices, see section 7.

6.3.3 GET

This operation is used for retrieving the status of an individual Twinning relationship.

6.3.3.1 Example 1: Retrieve status of an individual Twinning relationship (Informative)

6.3.3.1.1 Request

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/100/status  HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d77-4aa6-a404-2bc19b5cf999
Accept: application/xml
```

6.3.3.1.2 Response

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 07:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<twinning:status xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <twinningStatus>On</twinningStatus>
</twinning:status>
```

6.3.4 PUT

This operation is used to change (Toggle On or Off) the status of a Twinning resource and effective turn on/off the Twinning relationship.

In inter-MNO scenarios one server SHALL invoke the PUT method onto the other device’s server in order to update the twinning status of the corresponding twinning resource accordingly.

6.3.4.1 Example 1: Toggle Off a Twinning relationship (Informative)

6.3.4.1.1 Request

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/100/status HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:status xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <twinningStatus>Off</twinningStatus>
</twinning:status>
```

6.3.4.1.2 Response

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 08:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<twinning:status xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <twinningStatus>Off</twinningStatus>
</twinning:status>
```

6.3.4.2 Example 2: Toggle Off a Twinning relationship, response with HTTP 202 Accepted (Informative)

6.3.4.2.1 Request

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789/status HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:status xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <twinningStatus>Off</twinningStatus>
</twinning:status>
```

6.3.4.2.2 Response

```
HTTP/1.1 202 Accepted
Date: Tue, 03 Feb 2015 08:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<twinning:status xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <twinningStatus>Pending</twinningStatus>
</twinning:status>
```

6.3.5 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, PUT’ field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

6.3.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, PUT’ field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

6.4 Resource: Twinning user preference parameters

The resource used is:

//{serverRoot}/twinning/{apiVersion}/{userId}/twinningRelations/{twinningId}/userPreference

This resource is used to manage the Twinning relationship’s parameters related to user preferences (e.g. user turns on/off being notified of Twinning events out of band via SMS).

6.4.1 Request URL variables

The following request URL variables are common for all HTTP methods:

Name	Description
serverRoot	Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI
apiVersion	Version of the API client wants to use. The value of this variable is defined in section 5.2
userId	Identifier of the user on whose behalf the application acts Examples: tel:+19585550100, acr:pseudonym123, sip:alice@example.com
twinningId	Identifier of the twinning relationship generated by the server during creation of a Twinning relationship

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.4.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to RESTful Network API for Twinning Devices, see section 7.

6.4.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: PUT’ field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

6.4.4 PUT

This operation is used to update user preference parameters such as Twinning relationship “name” and/or “outOfBandNotifyPrimaryUser”.

In inter-MNO scenarios one server SHALL invoke the PUT method onto the other device’s server in order to update the corresponding twinning resource accordingly.

6.4.4.1 Example 1: Rename a Twinning relationship (Informative)

6.4.4.1.1 Request

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789/userPreference HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:userPreference xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <name>MySportsCar</name>
</twinning:userPreference>
```

6.4.4.1.2 Response

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 08:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<twinning:userPreference xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <name>MySportsCar</name>
</twinning:userPreference>
```

6.4.4.2 Example 2: Rename a Twinning relationship and also turn on the 'Out-of-band' notifications to Primary user (Informative)

6.4.4.2.1 Request

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789/userPreference HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:userPreference xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <name>newWatch</name>
  <outOfBandNotifyPrimaryUser>true</outOfBandNotifyPrimaryUser>
</twinning:userPreference>
```

6.4.4.2.2 Response

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 08:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<twinning:userPreference xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <name>newWatch</name>
  <outOfBandNotifyPrimaryUser>true</outOfBandNotifyPrimaryUser>
</twinning:userPreference>
```

6.4.4.3 Example 3: Turn on the ‘Out-of-band’ notifications to Primary user and set the Primary user’s notification endpoint (Informative)

In this example, the service provider’s policy allows setting the outOfBandNotifyPrimaryUserId parameter to a random (non-verifiable) value other than the Primary’s MSISDN.

6.4.4.3.1 Request

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789/userPreference HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:userPreference xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <outOfBandNotifyPrimaryUser>true</outOfBandNotifyPrimaryUser>
  <outOfBandNotifyPrimaryUserId>tel:+19585550109</outOfBandNotifyPrimaryUserId>
</twinning:userPreference>
```

6.4.4.3.2 Response

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 08:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<twinning:userPreference xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <outOfBandNotifyPrimaryUser>true</outOfBandNotifyPrimaryUser>
  <outOfBandNotifyPrimaryUserId>tel:+19585550109</outOfBandNotifyPrimaryUserId>
</twinning:userPreference>
```

6.4.4.4 Example 4: Turn on the ‘Out-of-band’ notifications to Primary user and set the Primary user’s notification endpoint, failure due to not allowed value (Informative)

In this example, due to service provider’s policy, setting the outOfBandNotifyPrimaryUserId parameter to a random phone number which can’t be verified to belong to the Primary user causes a policy error.

6.4.4.4.1 Request

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550107/twinningRelations/777/userPreference HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:userPreference xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <outOfBandNotifyPrimaryUser>true</outOfBandNotifyPrimaryUser>
  <outOfBandNotifyPrimaryUserId>tel:+19585550109</outOfBandNotifyPrimaryUserId>
</twinning:userPreference>
```

6.4.4.4.2 Response

```
HTTP/1.1 403 Forbidden
Date: Tue, 03 Feb 2015 04:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
   <policyException>
    <messageId>POL0002</messageId>
    <text>Privacy verification failed for address %1, request is refused</text>
    <variables>+19585550109</variables>
  </policyException>
</common:requestError>
```

6.4.4.5 Example 5: Rename a Twinning relationship, failure due to name duplication (Informative)

In this example, setting the name parameter fails as there is another existing Twinning relationship with the same name for the given Primary Device.

6.4.4.5.1 Request

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/788/userPreference HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:userPreference xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <name>MySportsCar</name>
</twinning:userPreference>
```

6.4.4.5.2 Response

```
HTTP/1.1 400 Bad request
Date: Tue, 03 Feb 2015 04:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <serviceException>
      <messageId>SVC2004</messageId>
     <text>Invalid input value for %1, %2: %3</text>
     <variables>element</variables>
     <variables>name</variables>
     <variables>duplicated name</variables>
  </serviceException>
</common:requestError>
```

6.4.5 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: PUT’ field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

6.4.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: PUT’ field in the response as per sections 6.5.5 and 7.4.1 of [RFC 2616].

6.5 Resource: All subscriptions to Twinning notifications

The resource used is:

//{serverRoot}/twinning/{apiVersion}/{userId}/subscriptions

This resource is used to manage subscriptions to event notifications related to Twinning devices.

This resource can be used in conjunction with a Client-side Notification URL, or in conjunction with a Server-side Notification URL. In this latter case, the application MUST first create a Notification Channel (see [REST_NetAPI_NotificationChannel]) before creating a subscription.

6.5.1 Request URL variables

The following request URL variables are common for all HTTP methods:

Name	Description
serverRoot	Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI
apiVersion	Version of the API client wants to use. The value of this variable is defined in section 5.2
userId	Identifier of the user on whose behalf the application acts Examples: tel:+19585550100, acr:pseudonym123, sip:alice@example.com

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.5.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to RESTful Network API for Twinning devices, see section 7.

6.5.3 GET

This operation is used for reading the list of active notification subscriptions. Note that only the subscriptions this client is authorised to access SHALL be included in the list.

6.5.3.1 Example: Reading all active subscriptions (Informative)

Application client reads all active subscriptions.

6.5.3.1.1 Request

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions HTTP/1.1
Accept: application/xml
Host: example.com
```

6.5.3.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnnn
Date: Tue, 03 Feb 2015 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinningSubscriptionList xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
   <subscription>
      <callbackReference>
         <notifyURL>http://applicationClient.example.com/twinning/notifications/77777</notifyURL>
         <callbackData>abcd</callbackData>
      </callbackReference>
      <duration>6300</duration>
      <clientCorrelator>12345</clientCorrelator>
      <resourceURL>http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001</resourceURL>
   </subscription>
   <resourceURL>http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions</resourceURL>
</twinning:twinningSubscriptionList>
```

6.5.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET, POST' field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.5.5 POST

This operation is used to create a new subscription for notifications related to Twinning devices events.

The notifyURL in the callbackReference either contains the Client-side Notification URL (as defined by the client) or the Server-side Notification URL (as obtained during the creation of the Notification Channel [REST_NetAPI_NotificationChannel]).

6.5.5.1 Example 1: Creating a new subscription, response with copy of created resource (Informative)

Application client creates a subscription.

6.5.5.1.1 Request

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinningSubscription xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
   <callbackReference>
      <notifyURL>http://applicationClient.example.com/twinning/notifications/77777</notifyURL>
      <callbackData>abcd</callbackData>
   </callbackReference>
   <duration>7200</duration>
   <clientCorrelator>12345</clientCorrelator>
</twinning:twinningSubscription>
```

6.5.5.1.2 Response

```
HTTP/1.1 201 Created
Content-Type: application/xml
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001
Content-Length: nnnn
Date: Tue, 03 Feb 2015 20:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinningSubscription xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
   <callbackReference>
      <notifyURL>http://applicationClient.example.com/twinning/notifications/77777</notifyURL>
      <callbackData>abcd</callbackData>
   </callbackReference>
   <duration>7200</duration>
   <clientCorrelator>12345</clientCorrelator>
   <resourceURL>http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001</resourceURL>
</twinning:twinningSubscription>
```

6.5.5.2 Example 2: Creating a new subscription, response with location of created resource (Informative)

Application client creates a subscription.

6.5.5.2.1 Request

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
```

```
<twinning:twinningSubscription xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <callbackReference>
    <notifyURL>http://applicationClient.example.com/twinning/notifications/77777</notifyURL>
    <callbackData>abcd</callbackData>
  </callbackReference>
  <duration>7200</duration>
  <clientCorrelator>12345</clientCorrelator>
</twinning:twinningSubscription>
```

6.5.5.2.2 Response

```
HTTP/1.1 201 Created
Content-Type: application/xml
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001
Content-Length: nnnn
Date: Tue, 03 Feb 2015 20:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<common:resourceReference xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <resourceURL>http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001</resourceURL>
</common:resourceReference>
```

6.5.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: GET, POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.6 Resource: Individual subscription to Twinning notifications

The resource used is:

//{serverRoot}/twinning/{apiVersion}/{userId}/subscriptions/{subscriptionId}

This resource is used to manage an individual event subscription. This resource can be used in conjunction with a Client-side Notification URL, or in conjunction with a Server-side Notification URL. In this latter case, the application MUST first create a Notification Channel (see [REST_NetAPI_NotificationChannel]) before creating a subscription.

6.6.1 Request URL variables

The following request URL variables are common for all HTTP methods:

Name	Description
serverRoot	Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI
apiVersion	Version of the API client wants to use. The value of this variable is defined in section 5.2
userId	Identifier of the user on whose behalf the application acts Examples: tel:+19585550100, acr:pseudonym123, sip:alice@example.com
subscriptionId	Identifier of the subscription

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.6.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to the RESTful Network API for Twinning devices, see section 7.

6.6.3 GET

This operation is used for reading an individual subscription.

6.6.3.1 Example 1: Reading an individual subscription (Informative)

Application client reads a subscription.

6.6.3.1.1 Request

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001 HTTP/1.1
Accept: application/xml
Host: example.com
```

6.6.3.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnnn
Date: Wed, 15 Jan 2014 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinningSubscription xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
    <callbackReference>
        <notifyURL>http://applicationClient.example.com/twinning/notifications/77777</notifyURL>
        <callbackData>abcd</callbackData>
    </callbackReference>
    <duration>7200</duration>
    <clientCorrelator>12345</clientCorrelator>
    <resourceURL>http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001</resourceURL>
</twinning:twinningSubscription>
```

6.6.3.2 Example 2: Retrieve information about a non-existent individual subscription (Informative)

6.6.3.2.1 Request

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550100/subscriptions/sub6699 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/xml
```

6.6.3.2.2 Response

```
HTTP/1.1 404 Not Found
Content-Type: application/xml
Content-Length: nnnn
Date: Fri, 11 Apr 2014 17:51:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
<common:requestError xmlns:common="urn:oma:xml:rest:netapi:common:1">
  <link rel="TwinningSubscription"
    href="http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550100/subscriptions/sub6699"/>
  <serviceException>
    <messageId>SVC0004</messageId>
    <text>No valid addresses provided in message part %1</text>
    <variables>Request-URI</variables>
  </serviceException>
</common:requestError>
```

6.6.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET, DELETE' field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.6.5 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: GET, DELETE' field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

6.6.6 DELETE

This operation is used to cancel a subscription and to stop corresponding notifications.

6.6.6.1 Example: Cancelling a subscription (Informative)

Application client cancels a subscription.

6.6.6.1.1 Request

```
DELETE /exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001 HTTP/1.1
Accept: application/xml
Host: example.com
```

6.6.6.1.2 Response

```
HTTP/1.1 204 No Content
Date: Wed, 15 Jan 2014 18:51:59 GMT
```

6.7 Resource: Subscription duration

The resource used is:

//{serverRoot}/twinning/{apiVersion}/{userId}/subscriptions/{subscriptionId}/duration

This resource is used to update (i.e. refresh) the duration of a subscription.

6.7.1 Request URL variables

The following request URL variables are common for all HTTP methods:

Name	Description
serverRoot	Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI
apiVersion	Version of the API client wants to use. The value of this variable is defined in section 5.2
userId	Identifier of the user on whose behalf the application acts Examples: tel:+19585550100, acr:pseudonym123, sip:alice@example.com
subscriptionId	Identifier of the subscription

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.7.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to RESTful Network API for Twinning Devices, see section 7.

6.7.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: PUT’ field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

6.7.4 PUT

This operation is used to update the subscription’s duration.

6.7.4.1 Refresh an existing subscription (Informative)

6.7.4.1.1 Request

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001/duration HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:subscriptionDuration xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <duration>555555</duration>
</twinning:subscriptionDuration >
```

6.7.4.1.2 Response

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 08:51:59 GMT
Content-Type: application/xml
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<twinning:subscriptionDuration xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <duration>555555</duration>
</twinning:subscriptionDuration >
```

6.7.5 POST

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: PUT' field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

6.8 Resource: Client notification about Twinning events

This resource is a callback URL provided by the client for notifications about Twinning events.

The RESTful Twinning devices API does not make any assumption about the structure of this URL. If this URL is a Client-side Notification URL, the server will POST notifications directly to it. If this URL is a Server-side Notification URL, the server uses it to determine the address of the Notification Server to which the notifications will subsequently be POSTed. The way the server determines the address of the Notification Server is out of scope of this specification.

Note: In the case when the client has set up a Notification Channel to obtain the notifications, the client needs to use the mechanisms described in [REST_NetAPI_NotificationChannel], instead of the mechanism described below in section 6.8.5.

The following table gives detailed information about Twinning notification.

EventType	Notification Root Element Type	Notification sent to	Response to Notification	Link rel	Link href Base URL: //{serverRoot}/twinning/{apiVersion}/{userId}
Activated	TwinningEventNotification	client	n/a	TwinningSubscription	/subscriptions/{subscriptionId}
Deactivated	TwinningEventNotification	client	n/a	TwinningSubscription	/subscriptions/{subscriptionId}
ToggledOn	TwinningEventNotification	client	n/a	TwinningSubscription	/subscriptions/{subscriptionId}
ToggledOff	TwinningEventNotification	client	n/a	TwinningSubscription	/subscriptions/{subscriptionId}

Table 1: Twinning event notification

6.8.1 Request URL variables

Client provided if any.

6.8.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to the RESTful Network API for Twinning devices, see section 7.

6.8.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.8.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.8.5 POST

This operation is used to notify the client about Twinning events.

6.8.5.1 Example 1: Notify a client about Twinning status change (Informative)

In this example, the application client is notified asynchronously that the Twinning relationship status has changed to “On”. In this example it is assumed that the previous Twinning status was “Pending” and it just changed to “On” due to Twinning activation successful completion.

6.8.5.1.1 Request

```
POST /twinning/notifications/77777 HTTP/1.1
Accept: application/xml
Content-Type: application/xml
Host: applicationClient.example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinningEventNotification xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
   <callbackData>12345</callbackData>
   <eventType>Activated</eventType>
   <twinningResourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/2000</twinningResourceURL>
   <link rel="TwinningSubscription"
          href="http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550100/subscriptions/sub001"/>
</twinning:twinningEventNotification>
```

6.8.5.1.2 Response

```
HTTP/1.1 204 No Content
Date: Fri, 28 Jun 2013 17:51:59 GMT
```

6.8.5.2 Example 2: Notify a client about Twinning status toggle event (Informative)

In this example, the application client is notified asynchronously of a Twinning toggle event. It is assumed that the Twinning status used to be “On” prior to this event notification.

6.8.5.2.1 Request

```
POST /twinning/notifications/77777 HTTP/1.1
Accept: application/xml
Content-Type: application/xml
Host: applicationClient.example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:twinningEventNotification xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <callbackData>12345</callbackData>
  <eventType>ToggledOff</eventType>
  <twinningResourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/1800</twinningResourceURL>
  <link rel="TwinningSubscription"
        href="http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550100/subscriptions/sub001"/>
</twinning:twinningEventNotification>
```

6.8.5.2.2 Response

```
HTTP/1.1 204 No Content
Date: Fri, 28 Jun 2013 17:51:59 GMT
```

6.8.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per section 6.5.5 and 7.4.1 of [RFC7231].

6.9 Resource: Inter-MNO Twinning activation code

The resource used is:

//{serverRoot}/twinning/{apiVersion}/interMnoActivationCode

In inter-MNO Twinning scenarios, this resource is used by one MNO to acquiring a new Twinning Activation Code from another MNO.

6.9.1 Request URL variables

The following request URL variables are common for all HTTP methods:

Name	Description
serverRoot	Server base url: hostname+port+base path. Port and base path are OPTIONAL. Example: example.com/exampleAPI
apiVersion	Version of the API client wants to use. The value of this variable is defined in section 5.1

See section 6 for a statement on the escaping of reserved characters in URL variables.

6.9.2 Response Codes and Error Handling

For HTTP response codes, see [REST_NetAPI_Common].

For Policy Exception and Service Exception fault codes applicable to RESTful Network API for Twinning Devices, see section 7.

6.9.3 GET

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

6.9.4 PUT

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the 'Allow: POST' field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

6.9.5 POST

In an inter-MNO Twinning scenario (which is triggered by the Secondary Device's Twinning activation request), the Secondary Device's server uses this POST operation to ask the Primary Device's server for a Twinning Activation Code. See section 5.4.7 for further information.

6.9.5.1 Secondary Device's server requests a new Twinning Activation Code from the Primary Device's server (Informative)

6.9.5.1.1 Request

```
POST /exampleAPI/twinning/v1/interMnoActivationCode / HTTP/1.1
Content-Type: application/xml
Content-Length: nnnn
Accept: application/xml
Host: example.com

<?xml version="1.0" encoding="UTF-8"?>
<twinning:interMnoActivationCode xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <twinningResourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550103/twinningRelations/333</twinningResourceURL>
  <address>tel:+19585550103</address>
</twinning:interMnoActivationCode>
```

6.9.5.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: nnnn
Date: Tue, 03 Feb 2015 21:32:52 GMT

<?xml version="1.0" encoding="UTF-8"?>
<twinning:interMnoActivationCode xmlns:twinning="urn:oma:xml:rest:netapi:twinning:1">
  <twinningResourceURL>http://exampleAPI/twinning/v1/tel%3A%2B19585550103/twinningRelations/333</twinningResourceURL>
  <address>tel:+19585550103</address>
  <activationCode>
    <code>711711</code>
    <mno>
      <name>MNO-A</name>
      <ncc>310410</ncc>
    </mno>
   </activationCode>
</twinning:interMnoActivationCode>
```

6.9.6 DELETE

Method not allowed by the resource. The returned HTTP error status is 405. The server should also include the ‘Allow: POST’ field in the response as per sections 6.5.5 and 7.4.1 of [RFC7231].

7. Fault definitions

7.1 Service Exceptions

For common Service Exceptions refer to [REST_NetAPI_Common]. The following additional Service Exception codes are defined for the RESTful Twinning Devices API.

7.1.1 SVC1014: Undefined twinning role

Name	Description
MessageID	SVC1014
Text	Role is required to form a Twinning relationship.
Variables	None
HTTP status code(s)	400 Bad Request

7.2 Policy Exceptions

For common Policy Exceptions refer to [REST_NetAPI_Common]. The following additional Policy Exception codes are defined for the RESTful Twinning Devices API.

7.2.1 POL1037: Twinning role mixture not allowed

Name	Description
MessageID	POL1037
Text	A Twinned Primary Device is not allowed to take on a Secondary role in another Twinning relationship
Variables	None
HTTP status code(s)	403 Forbidden

7.2.2 POL1038: Twinning data update not allowed

Name	Description
MessageID	POL1038
Text	Updating Twinning data of an active Twinning relationship is not allowed
Variables	None
HTTP status code(s)	403 Forbidden

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
n/a	n/a	No prior version

A.2 Draft/Candidate Version 1.0 History

Document Identifier	Date	Sections	Description
Draft Versions: REST_NetAPI _Twinning-V1_0	22 Oct. 2014	All	Initial baseline
	22 Jan. 2015	2.2, 4, 4.1, 5, 5.1, 5.2	OMA-ARC-TWIN-2015-0001R01-CR_Progress_TS, OMA-ARC-TWIN-2015-0002R03-CR_Progress_TS_Section_5, OMA-ARC-TWIN-2015-0004R02-CR_Concepts_section
	04 Feb. 2015	5.3, 5.4, 6, 6.1	OMA-ARC-TWIN-2015-0007R03-CR_ResourceTree_and_dataTypes, OMA-ARC-TWIN-2015-0008R01-CR_TS_Sequence_Diagrams, OMA-ARC-TWIN-2015-0010R01-CR_XML_Examples, OMA-ARC-TWIN-2015-0011R01-CR_XML_Examples_ToggleAndSubscription, OMA-ARC-TWIN-2015-0012-CR_Enhanced_event_dataType
	05 Mar. 2015	All	OMA-ARC-TWIN-2015-0015R01-CR_initial_sections
	01 Apr. 2015	4.0, 5.1, 5.3, 5.4 & 6.6	OMA-ARC-TWIN-2015-0018R01-CR_Concept_section_edits, OMA-ARC-TWIN-2015-0019R02-CR_efficient_notif, OMA-ARC-TWIN-2015-0020-CR_Primary_VM_Address
	29 Apr 2015	Appendix D, 6.1, 6.2, 7, Appendix F, Appendix G, 5.3.2.4, 6.3, 6.6	OMA-ARC-TWIN-2015-0022-CR_add_error_examples, OMA-ARC-TWIN-2015-0023-CR_add_section_7, OMA-ARC-TWIN-2015-0024-CR_edit_auth_section, OMA-ARC-TWIN-2015-0025R02-CR_Primary_VM_Address_fix, OMA-ARC-TWIN-2015-0027-CR_typo_fix, OMA-ARC-TWIN-2015-0028-CR_JSON_examples, OMA-ARC-TWIN-2015-0029R01-CR_Notification_cleanup
	11 Jun 2015	5.2.1, 5.3.2.4, 5.4.6, 6.2.4, 6.2.5, 7.2, 6.2.3.3, 6.1.5.5, 6.7 Appendix D, Appendix G, Appendix B	OMA-ARC-TWIN-2015-0031-CR_Policy_error_Number_change OMA-ARC-TWIN-2015-0032R03-CR_POST_for_interMNO_update, OMA-ARC-TWIN-2015-0034R01-CR_Cleanups, OMA-ARC-TWIN-2015-0035-CR_Additional_JSON_examples, OMA-ARC-TWIN-2015-0036R01-CR_SCR_section, OMA-ARC-TWIN-2015-0037-CR_InterMnoActivationCode_examples, OMA-ARC-TWIN-2015-0039R02-CR_Appendix_G_update
	23 Jul 2015	5.2,x, 6.6 Appendix D,	OMA-ARC-TWIN-2015-0040R02-CR_refresh_subscription, OMA-ARC-TWIN-2015-0041R02-CR_editorials, OMA-ARC-TWIN-2015-0043R01-CR_outOfBand_notify, OMA-ARC-TWIN-2015-0046-CR_VMaddress_in_example
	28 Jul 2015	5.,x, 6.x Appendix D,	OMA-ARC-TWIN-2015-0044R02-CR_userPreference_update, OMA-ARC-TWIN-2015-0047-CR_interMNO_dataType_simplification, OMA-ARC-TWIN-2015-0048-CR_Optional_POST_Response

Document Identifier	Date	Sections	Description
	17 Aug 2015	Appendix G, Sections 6.x.x, 5.3.x.x, 5.4.x Appendix D,	OMA-ARC-TWIN-2015-0049R01-CR_Async_Response, OMA-ARC-TWIN-2015-0050-CR_Optional_role, OMA-ARC-TWIN-2015-0051-CR_Twinning_POST_datatype_simplification, OMA-ARC-TWIN-2015-0052-CR_outOfBandNotifyUserId, OMA-ARC-TWIN-2015-0053-CR_Seq_diagrams_addition, OMA-ARC-TWIN-2015-0055-CR_Appendix_G_new_resources, OMA-ARC-TWIN-2015-0056-CR_refresh_subscription_update
	01 Sep 2015	All	OMA-ARC-TWIN-2015-0057R02-CR_fix_parameter_name, OMA-ARC-TWIN-2015-0058-CR_updated_operationslist, OMA-ARC-TWIN-2015-0059R01-CR_SCR_table_update, OMA-ARC-TWIN-2015-0060R01-CR_Editorials, OMA-ARC-TWIN-2015-0061-CR_missing_JSON_examples, OMA-ARC-TWIN-2015-0062-CR_cleanups
	18 Nov 2015	All	OMA-ARC-TWIN-2015-0068R01-CR_Addressing_CONR_identified_issues, OMA-ARC-TWIN-2015-0069R01-CR_addressing_CONR_items, OMA-ARC-TWIN-2015-0070-CR_CONR_twinningUpdate_name_change, OMA-ARC-TWIN-2015-0071-CR_CONR_resourceURL_element, OMA-ARC-TWIN-2015-0072-CR_CONR_instruction_items, OMA-ARC-TWIN-2015-0073-CR_CONR_flows_update, OMA-ARC-TWIN-2015-0074-CR_CONR_new_POL2000_example, OMA-ARC-TWIN-2015-0081-CR_FFS_resoltution
Candidate Version: REST_NetAPI _Twinning-V1_0	15 Dec 2015	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2015-0216-INP_TWIN_V1_0_ERP_and_ETR_for_Candidate_approval

Appendix B. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [SCRRULES].

B.1 SCR for REST.TWINNING Server

Item	Function	Reference	Requirement
REST-TWINNING-SUPPORT-S-001-M	Support for the RESTful Twinning Devices API	5, 6	
REST-TWINNING-SUPPORT-S-002-M	Support for the XML request & response format	6	
REST-TWINNING-SUPPORT-S-003-M	Support for the JSON request & response format	6	

B.1.1 SCR for REST.TWINNING.Relationships Server

Item	Function	Reference	Requirement
REST-TWINNING-RELATIONSHIPS-S-001-M	Support for Twinning devices	6.1	
REST-TWINNING-RELATIONSHIPS-S-002-M	Retrieve existing Twinning relationships a device is involved in including retrieving instructions on Twinning activation – GET	6.1.3	
REST-TWINNING-RELATIONSHIPS-S-003-M	Support for Twinning relationship creation - POST	6.1.5	

B.1.2 SCR for REST.TWINNING.ARelationship Server

Item	Function	Reference	Requirement
REST-TWINNING-ARELATIONSHIP-S-001-M	Support for managing individual Twinning relationship	6.2	
REST-TWINNING-ARELATIONSHIP-S-002-M	Retrieve Twinning relationship information – GET	6.2.3	
REST-TWINNING-ARELATIONSHIP-S-003-O	Update/activate an interMNO “pending” Twinning relationship – POST	6.2.5	
REST-TWINNING-ARELATIONSHIP-S-004-M	Delete Twinning relationship – DELETE	6.2.6	

B.1.3 SCR for REST.TWINNING.ARelationship.Status Server

Item	Function	Reference	Requirement
REST-TWINNING-ARELATIONSHIP-STATUS-S-001-M	Support for managing status of Twinning relationship	6.3	
REST-TWINNING-ARELATIONSHIP-STATUS-S-002-O	Retrieve status of Twinning relationship – GET	6.3.3	
REST-TWINNING-ARELATIONSHIP-STATUS-S-003-M	Toggle Off/On a Twinning relationship – PUT	6.3.4	

B.1.4 SCR for REST.TWINNING.ARelationship.UserPref Server

Item	Function	Reference	Requirement
REST-TWINNING-ARELATIONSHIP-USERPERF-S-001-M	Support for managing user preference paramters of Twinning relationship	6.4	
REST-TWINNING-ARELATIONSHIP-USERPERF-S-002-M	Update the user perference parameters of a Twinning relationship – PUT	6.4.4	

B.1.5 SCR for REST.TWINNING.Subscriptions Server

Item	Function	Reference	Requirement
REST-TWINNING-SUBSCRIPTIONS-S-001-M	Support for subscriptions to Twinning events	6.5	
REST-TWINNING-SUBSCRIPTIONS-S-002-O	Read the list of active subscriptions to Twinning event notifications – GET	6.5.3	
REST-TWINNING-SUBSCRIPTIONS-S-003-M	Create new subscription to Twinning event notifications – POST	6.5.5	

B.1.6 SCR for REST.TWINNING.IndSubscription Server

Item	Function	Reference	Requirement
REST-TWINNING-INDSUBSCR-S-001-O	Support for access to an individual subscription to Twinning event notifications	6.6	REST-TWINNING-INDSUBSCR-S-002-O AND REST-TWINNING-INDSUBSCR-S-003-O
REST-TWINNING-INDSUBSCR-S-002-O	Read an individual subscription to Twinning event notifications – GET	6.6.3	
REST-TWINNING-INDSUBSCR-S-003-O	Cancel subscription and stop corresponding notifications – DELETE	6.6.6	

B.1.7 SCR for REST.TWINNING.IndSubscription.Duration Server

Item	Function	Reference	Requirement
REST-TWINNING-INDSUBSCR-DURATION-S-001-O	Support for refreshing an individual subscription to Twinning event notifications	6.7	REST-TWINNING-INDSUBSCR-DURATION-S-002-O
REST-TWINNING-INDSUBSCR-DURATION-S-002-O	Extend the duration of an individual subscription to Twinning event notifications – PUT	6.7.4	

B.1.8 SCR for REST.TWINNING.Notifications Server

Item	Function	Reference	Requirement
REST-TWINNING-NOTIF-S-001-M	Support for notifications about Twinning events	6.7	
REST-TWINNING-NOTIF-S-002-M	Notifications about Twinning events – POST	6.8.5	

B.1.9 SCR for REST.TWINNING.InterMNO Server

Item	Function	Reference	Requirement
REST-TWINNING-INTERMNO-S-001-O	Support for InterMNO Twinning activation	6.9	REST-TWINNING-INTERMNO-S-002-O
REST-TWINNING-INTERMNO-S-002-O	Create new Activation Twinning Code – POST	6.9.5	

Appendix C. Application/x-www-form-urlencoded Request Format for POST Operations (Normative)

This specification does not define any API request based on application/x-www-form-urlencoded MIME type.

Appendix D. JSON examples (Informative)

JSON (JavaScript Object Notation) is a Light-weight, text-based, language-independent data interchange format. It provides a simple means to represent basic name-value pairs, arrays and objects. JSON is relatively trivial to parse and evaluate using standard JavaScript libraries, and hence is suited for REST invocations from browsers or other processors with JavaScript engines. Further information on JSON can be found at [RFC7159].

The following examples show the request and response for various operations using the JSON data format. The examples follow the XML to JSON serialization rules in [REST_NetAPI_Common]. A JSON response can be obtained by using the content type negotiation mechanism specified in [REST_NetAPI_Common].

For full details on the operations themselves please refer to the section number indicated.

D.1 Retrieve a Primary Device's existing Twinning relationship (section 6.1.3.1)

Request:

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations  HTTP/1.1
Host: example.com
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 02:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"twinningList": {
   "resourceURL": "http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations",
   "twinning": [
    {
        "name": "MyCar",
        "otherDevice": {
            "address": "tel:+19585550101",
            "mno": {
                "name": "MNO-A",
                "ncc": "310410"
            },
            "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550101/twinningRelations/1900"
        },
        "ownAddress": "tel:+19585550100",
        "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/1800",
        "role": "Primary",
        "twinningStatus": "On"
    },
    {
        "name": "MyWatch",
        "otherDevice": {
            "address": "tel:+19585550102",
            "mno": {
                "name": "MNO-A",
                "ncc": "310410"
            },
            "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550102/twinningRelations/222"
        },
        "ownAddress": "tel:+19585550100",
        "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/111",
        "role": "Primary",
        "twinningStatus": "Off"
    }
]}}
```

D.2 Retrieve a Secondary Device's existing Twinning relationship using "auth" keyword (section 6.1.3.2)

Request:

```
GET /exampleAPI/twinning/v1/acr%3Aauth/twinningRelations  HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d77-4aa6-a404-2bc19b5cf333
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 02:54:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"twinningList": {
   "resourceURL": "http://example.com/exampleAPI/twinning/v1/acr%3Aauth/twinningRelations",
   "twinning": {
    "name": "MyCar",
    "otherDevice": {
        "address": "tel:+19585550100",
        "mno": {
            "name": "MNO-A",
            "ncc": "310410"
        },
        "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/1900"
    },
    "ownAddress": "tel:+19585550101",
    "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550101/twinningRelations/1800",
    "role": "Secondary",
    "twinningStatus": "On"
}}}
```

D.3 Retrieve Twinning instruction from a Secondary Device (section 6.1.3.3)

Request:

```
GET /exampleAPI/twinning/v1/acr%3Aauth/twinningRelations?instruction=ForSecondary  HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d77-4aa6-a404-2bc19b5cf333
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 02:54:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"twinningList": {
   "resourceURL": "http://example.com/exampleAPI/twinning/v1/acr%3Aauth/twinningRelations",
   "instruction": {
    "activationInstruction": "If the Primary Device with which this device is intended to be twinned is served by a different mobile service
provider please select the Primary Device's mobile service provider from the list",
    "mno": [
       {
          "name": "MNO-A",
          "ncc": "310410"
       },
       {
          "name": "MNO-B",
          "ncc": "23415"
       },
       {
          "name": "MNO-C",
          "ncc": "310260"
       },
       {
          "name": "MNO-D Wireless",
          "ncc": "23208"
       }
    ]
}}}
```

D.4 Retrieve Twinning instruction from a Primary Device (section 6.1.3.4)

Request:

```
GET /exampleAPI/twinning/v1/acr%3Aauth/twinningRelations?instruction=ForPrimary  HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d77-4aa6-a404-2bc19b511111
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 02:54:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"twinningList": {
   "resourceURL": "http://example.com/exampleAPI/twinning/v1/acr%3Aauth/twinningRelations",
    "instruction": {"activationInstruction": "please enter an activation code acquired from the Secondary Device with which this device is
intended to be twinned with"},
    "twinning": [
        {
            "name": "MyCar",
            "otherDevice": {
                "address": "tel:+19585550101",
                "mno": {
                    "name": "MNO-A",
                    "ncc": "310410"
                },
                "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550101/twinningRelations/1900"
            },
            "ownAddress": "tel:+19585550100",
            "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/1800",
            "role": "Primary",
            "twinningStatus": "On"
        },
        {
            "name": "MyWatch",
            "otherDevice": {
                "address": "tel:+19585550102",
                "mno": {
                    "name": "MNO-A",
                    "ncc": "310410"
                },
                "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550102/twinningRelations/222"
            },
            "ownAddress": "tel:+19585550100",
            "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/111",
            "role": "Primary",
            "twinningStatus": "Off"
        }
    ]
}}
```

D.5 A Secondary Device initiates a new Twinning relationship (section 6.1.5.1)

Request:

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550103/twinningRelations HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"twinning": {"role": "Secondary"}}
```

Response:

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550103/twinningRelations/333
Content-Type: application/json
Content-Length: nnnn
Date: Tue, 03 Feb 2015 21:32:52 GMT

{"twinning": {
    "activationCode": {
        "code": "711711",
        "mno": {
            "name": "MNO-A",
            "ncc": "310410"
        }
    },
    "ownAddress": "tel:+19585550103",
    "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550103/twinningRelations/333",
    "role": "Secondary",
    "twinningStatus": "Pending"
}}
```

D.6 A Primary Device initiates and activates a new Twinning relationship (section 6.1.5.2)

Request:

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"twinning": {
    "activationCode": {"code": "711711"},
    "name": "FamilySUV",
    "role": "Primary"
}}
```

Response:

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/2000
Content-Type: application/json
Content-Length: nnnn
Date: Tue, 03 Feb 2015 21:32:52 GMT

{"twinning": {
    "activationCode": {
        "code": "711711",
        "mno": {
            "name": "MNO-A",
            "ncc": "310410"
        }
    },
    "name": "FamilySUV",
    "otherDevice": {
        "address": "tel:+19585550103",
        "mno": {
            "name": "MNO-A",
            "ncc": "310410"
        },
        "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550103/twinningRelations/333"
    },
    "ownAddress": "tel:+19585550100",
    "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/2000",
    "role": "Primary",
    "twinningStatus": "On"
}}
```

D.7 A Primary Device initiates and activates a new Twinning relationship, response with HTTP 202 Accepted (section 6.1.5.3)

Request:

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"twinning": {
    "activationCode": {"code": "711711"},
    "name": "FamilySUV",
    "role": "Primary"
}}
```

Response:

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/2000
Content-Type: application/json
Content-Length: nnnn
Date: Tue, 03 Feb 2015 21:32:52 GMT

{"twinning": {
    "activationCode": {
        "code": "711711",
        "mno": {
            "name": "MNO-A",
            "ncc": "310410"
        }
    },
    "name": "FamilySUV",
    "otherDevice": {
        "address": "tel:+19585550103",
        "mno": {
            "name": "MNO-A",
            "ncc": "310410"
        },
        "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550103/twinningRelations/333"
    },
    "ownAddress": "tel:+19585550100",
    "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/2000",
    "role": "Primary",
    "twinningStatus": "Pending"
}}
```

D.8 A Primary Device initiates and activates a new Twinning relationship, response with location of created resource (section 6.1.5.3)

Request:

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"twinning": {
    "activationCode": {"code": "711711"},
    "name": "FamilySUV",
    "role": "Primary"
}}
```

Response:

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/2000
Content-Type: application/json
Content-Length: nnnn
Date: Tue, 03 Feb 2015 21:32:52 GMT

{"resourceReference": {
    "resourceURL": "http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/2000"
}}
```

D.9 A Secondary Device initiates a new inter-MNO Twinning relationship (section 6.1.5.3)

Request:

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"twinning": {
    "primaryMno": {
        "name": "MNO-A",
        "ncc": "310410"
    },
    "role": "Secondary"
}}
```

Response:

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789
Content-Type: application/json
Content-Length: nnnn
Date: Tue, 03 Feb 2015 21:32:52 GMT

{"twinning": {
    "activationCode": {
        "code": "123123",
        "mno": {
            "name": "MNO-A",
            "ncc": "310410"
        }
    },
    "ownAddress": "tel:+19585550104",
    "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789",
    "role": "Secondary",
    "twinningStatus": "Pending"
}}
```

D.10 A Primary Device initiates and activates a new inter-MNO Twinning relationship (section 6.1.5.6)

Request:

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"twinning": {
    "activationCode": {"code": "123123"},
    "name": "ConnectedCar",
    "role": "Primary"
}}
```

Response:

```
HTTP/1.1 201 Created
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/100
Content-Type: application/json
Content-Length: nnnn
Date: Tue, 03 Feb 2015 21:32:52 GMT

{"twinning": {
    "activationCode": {
        "code": "123123",
        "mno": {
            "name": "MNO-A",
            "ncc": "310410"
        }
    },
    "name": "ConnectedCar",
    "otherDevice": {
        "address": "tel:+19585550104",
        "mno": {
            "name": "MNO-B",
            "ncc": "310260"
        },
        "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789"
    },
    "ownAddress": "tel:+19585550100",
    "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/100",
    "role": "Primary",
    "twinningStatus": "On"
}}
```

D.11 A Primary Device initiates Twinning relationship activation, failure due to an expired activation code (section 6.1.5.7)

Request:

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"twinning": {
    "activationCode": {"code": "711711"},
    "name": "FamilySUV",
    "role": "Primary"
}}
```

Response:

```
HTTP/1.1 400 Bad Request
Date: Tue, 03 Feb 2015 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"requestError": {"serviceException": {
    "messageId": "SVC2004",
    "text": "Invalid input value for %1 %2: %3",
    "variables": [
        "activationCode",
        "711711",
        "has expired"
    ]
}}}
```

D.12 A Secondary Device initiates Twinning relationship activation, failure due to undefined twinning role (section 6.1.5.8)

Request:

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"twinning": null }
```

Response:

```
HTTP/1.1 400 Bad Request
Date: Tue, 03 Feb 2015 12:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"requestError": {"serviceException": {
   "messageId": "SVC1014",
   "text": "Role is required to form a Twinning relationship"
}}}
```

D.13 A Primary device initiates Twinning relationship activation, failure due to a policy error such as “maximum number of twinning relationships reached” (section 6.1.5.9)

Request:

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"twinning": {
   "activationCode": {"code": "713713"},
   "name": "sportsBand",
   "role": "Primary"
}}
```

Response:

```
HTTP/1.1 403 Forbidden
Date: Thu, 20 Nov 2014 21:51:51 GMT
Content-Type: application/json
Content-Length: nnnn

{"requestError": {"policyException": {
   "messageId": "POL2000",
   "text": "The following policy error occurred: %1. Error code is %2",
   "variables": [
      "Maximum number of Twinning relationships reached. Delete an existing Twinning relationship and try again",
      "E234"
   ]
}}}
```

D.14 Retrieve an individual Twinning relationship from Primary Device (section 6.2.3.1)

Request:

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/100  HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d77-4aa6-a404-2bc19b5cf999
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 04:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"twinning": {
    "activationCode": {
        "code": "123123",
        "mno": {
            "name": "MNO-A",
            "ncc": "310410"
        }
    },
    "name": "ConnectedCar",
    "otherDevice": {
        "address": "tel:+19585550104",
        "mno": {
            "name": "MNO-B",
            "ncc": "310260"
        },
        "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789"
    },
    "ownAddress": "tel:+19585550100",
    "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/100",
    "role": "Primary",
    "twinningStatus": "On"
}}
```

D.15 Retrieve an individual Twinning relationship from Secondary Device (section 6.2.3.2)

Request:

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789  HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d77-4aa6-a404-2bc19b5cf
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 04:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"twinning": {
    "activationCode": {
        "code": "123123",
        "mno": {
            "name": "MNO-A",
            "ncc": "310410"
        }
    },
    "ownAddress": "tel:+19585550104",
    "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789",
    "role": "Secondary",
    "twinningStatus": "Pending"
}}
```

D.16 Retrieve information about a non-existent individual Twinning relationship (section 6.2.3.3)

Request:

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/399621  HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d77-4aa6-a404-2bc19b5cf
Accept: application/json
HTTP/1.1 404 Not Found
```

Response:

```
Content-Type: application/json
Content-Length: nnnn
Date: Fri, 11 Apr 2014 17:51:59 GMT

{"requestError": {"serviceException": {
    "messageId": "SVC0004",
    "text": "No valid addresses provided in message part %1",
    "variables": "Request-URI"
}}}
```

D.17 Activate an inter-MNO “Pending” Twinning relationship (section 6.2.5.1)

Request:

```
POST  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789 HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

{"interMnoTwinningActivation": {
    "authCode": "6d0d-4aa6-a404",
    “primaryDeviceVMAddress": "tel:+19585550199",
    "name": "ConnectedCar",
    "otherDevice": {
        "address": "tel:+19585550100",
        "mno": {
            "name": "MNO-A",
            "ncc": "310410"
        },
        "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/100"
    }
}}
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 04:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"twinning": {
    "activationCode": {
        "code": "123123",
        "mno": {
            "name": "MNO-A",
            "ncc": "310410"
        }
    },
    "authCode": "6d0d-4aa6-a404",
    “primaryDeviceVMAddress": "tel:+19585550199",
    "name": "ConnectedCar",
    "otherDevice": {
        "address": "tel:+19585550100",
        "mno": {
            "name": "MNO-A",
            "ncc": "310410"
        },
        "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/100"
    },
    "ownAddress": "tel:+19585550104",
    "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789",
    "role": "Secondary",
    "twinningStatus": "On"
}}
```

D.18 Updating information about an already active Twinning relationship fails (section 6.2.5.2)

Request:

```
POST  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789 HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

{"interMnoTwinningActivation": {
    "authCode": "6d0d-4aa6-a404",
    "name": "ConnectedCar",
    "otherDevice": {
        "address": "tel:+19585550101",
        "mno": {
            "name": "MNO-A",
            "ncc": "310410"
        },
        "resourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550101/twinningRelations/101"
    }
}}
```

Response:

```
HTTP/1.1 403 Forbidden
Date: Tue, 03 Feb 2015 04:51:59 GMT
Content-Type: application/json
Content-Length: nnnn


{"requestError": {"policyException": {
    "messageId": "POL1038",
    "text": "Updating Twinning data of an active Twinning relationship is not allowed"
}}}
```

D.19 Delete a Twinning relationship (section 6.2.6.1)

Request:

```
DELETE /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
```

Response:

```
HTTP/1.1 204 No Content
Date: Tue, 03 Feb 2015 06:55:59 GMT
```

D.20 Retrieve status of an individual Twinning relationship (section 6.3.3.1)

Request:

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/100/status  HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d77-4aa6-a404-2bc19b5cf999
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
Date: Tue, 03 Feb 2015 17:51:59 GMT

{"status": {"twinningStatus": "On"}}
```

D.21 Toggle Off a Twinning relationship (section 6.3.4.1)

Request:

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/100/status HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

{"status": {"twinningStatus": "Off"}}
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 08:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"status": {"twinningStatus": "Off"}}
```

D.22 Toggle Off a Twinning relationship, response with HTTP 202 Accepted (section 6.3.4.1)

Request:

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789/status HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

{"status": {"twinningStatus": "Off"}}
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 08:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"status": {"twinningStatus": "Pending"}}
```

D.23 Rename a Twinning relationship (section 6.4.4.1)

Request:

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789/userPreference HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

{"userPreference": {"name": "MySportsCar"}}
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 08:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"userPreference": {
  "name": "MySportsCar"
    }
}
```

D.24 Rename a Twinning relationship and also turn 'Out-of-band' notifications to Primary user on (section 6.4.4.2)

Request:

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789/userPreference HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

{"userPreference": {
   "name": "newWatch",
   “outOfBandNotifyPrimaryUser”: “true”
    }
}
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 08:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"userPreference": {
   "name": "newWatch",
   “outOfBandNotifyPrimaryUser”: “true”
    }
}
```

D.25 Turn on the 'Out-of-band' notifications to Primary user and set the Primary user's notification endpoint (section 6.4.4.3)

Request:

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789/userPreference HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

{"userPreference": {
   “outOfBandNotifyPrimaryUser”: “true”,
   “outOfBandNotifyPrimaryUserId”: “+19585550109”
    }
}
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 08:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"userPreference": {
   "outOfBandNotifyPrimaryUser": "true",
   "outOfBandNotifyPrimaryUserId": "+19585550109"
    }
}
```

D.26 Turn on the 'Out-of-band' notifications to Primary user and set the Primary user's notification endpoint, failure due to not allowed value (section 6.4.4.4)

Request:

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550107/twinningRelations/777/userPreference HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

{"userPreference": {
   "outOfBandNotifyPrimaryUser": "true",
   "outOfBandNotifyPrimaryUserId": "+19585550109"
    }
}
```

Response:

```
HTTP/1.1 403 Forbidden
Date: Tue, 03 Feb 2015 04:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"requestError": {"policyException": {
    "messageId": "POL0002",
    "text": "Privacy verification failed for address %1, request is refused",
     "variables": [ "+19585550109" ]

}}}
```

D.27 Rename a Twinning relationship, failure due to name duplication (section 6.4.4.5)

Request:

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/788/userPreference HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

{"userPreference": {"name": "MySportsCar"}}
```

Response:

```
HTTP/1.1 400 Bad request
Date: Tue, 03 Feb 2015 04:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"requestError": {"serviceException": {
    "messageId": "SVC2004",
    "text": "Invalid input value for %1, %2: %3",
     "variables": [ "element", “name”, “duplicated name” ]

}}}
```

D.28 Reading all active subscriptions (section 6.5.3.1)

Request:

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions HTTP/1.1
Accept: application/json
Host: example.com
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
Date: Tue, 03 Feb 2015 17:51:59 GMT

{"twinningSubscriptionList": {
    "resourceURL": "http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions",
    "subscription": {
        "callbackReference": {
            "callbackData": "abcd",
            "notifyURL": "http://applicationClient.example.com/twinning/notifications/77777"
        },
        "clientCorrelator": "12345",
        "duration": "6300",
        "resourceURL": "http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001"
    }
}}
```

D.29 Creating a new subscription, response with copy of created resource (section 6.5.5.1)

Request:

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"twinningSubscription": {
    "callbackReference": {
        "callbackData": "abcd",
        "notifyURL": "http://applicationClient.example.com/twinning/notifications/77777"
    },
    "clientCorrelator": "12345",
    "duration": "7200"
}}
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001
Content-Length: nnnn
Date: Tue, 03 Feb 2015 20:51:59 GMT

{"twinningSubscription": {
    "callbackReference": {
        "callbackData": "abcd",
        "notifyURL": "http://applicationClient.example.com/twinning/notifications/77777"
    },
    "clientCorrelator": "12345",
    "duration": "7200",
    "resourceURL": "http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001"
}}
```

D.30 Creating a new subscription, response with location of created resource (section 6.5.5.2)

Request:

```
POST /exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"twinningSubscription": {
    "callbackReference": {
        "callbackData": "abcd",
        "notifyURL": "http://applicationClient.example.com/twinning/notifications/77777"
    },
    "clientCorrelator": "12345",
    "duration": "7200"
}}
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001
Content-Length: nnnn
Date: Tue, 03 Feb 2015 20:51:59 GMT

{"resourceReference": {
    "resourceURL": "http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001"
}}
```

D.31 Reading an individual subscription (section 6.6.3.1)

Request:

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001 HTTP/1.1
Accept: application/json
Host: example.com
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
Date: Wed, 15 Jan 2014 17:51:59 GMT

{"twinningSubscription": {
    "callbackReference": {
        "callbackData": "abcd",
        "notifyURL": "http://applicationClient.example.com/twinning/notifications/77777"
    },
    "clientCorrelator": "12345",
    "duration": "7200",
    "resourceURL": "http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001"
}}
```

D.32 Retrieve information about a non-existent individual subscription (section 6.6.3.2)

Request:

```
GET /exampleAPI/twinning/v1/tel%3A%2B19585550100/subscriptions/sub6699 HTTP/1.1
Host: example.com
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Accept: application/json
```

Response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json
Content-Length: nnnn
Date: Fri, 11 Apr 2014 17:51:59 GMT

{"requestError": {
    "link": {
        "href": "http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550100/subscriptions/sub6699",
        "rel": "TwinningSubscription"
    },
    "serviceException": {
        "messageId": "SVC0004",
        "text": "No valid addresses provided in message part %1",
        "variables": "Request-URI"
    }
}}
```

D.33 Cancelling a subscription (section 6.6.6.1)

Request:

```
DELETE /exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001 HTTP/1.1
Accept: application/json
Host: example.com
```

Response:

```
HTTP/1.1 204 No Content
Date: Wed, 15 Jan 2014 18:51:59 GMT
```

D.34 Refresh an existing subscription (section 6.7.4.1)

Request:

```
PUT  /exampleAPI/twinning/v1/tel%3A%2B19585550101/subscriptions/sub001/duration HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Authorization: BEARER 08776724-6d0d-4aa6-a404-2bc19b5cf903
Host: example.com

{"subscriptionDuration": {"duration": "555555"}}
```

Response:

```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 08:51:59 GMT
Content-Type: application/json
Content-Length: nnnn

{"subscriptionDuration": {"duration": "555555"}}
```

D.35 Notify a client about Twinning status change (section 6.8.5.1)

Request:

```
POST /twinning/notifications/77777 HTTP/1.1
Accept: application/json
Content-Type: application/json
Host: applicationClient.example.com

{"twinningEventNotification": {
    "callbackData": "12345",
    "eventType": "Activated",
    "link": {
        "href": "http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550104/subscriptions/sub001",
        "rel": "TwinningSubscription"
    },
    "twinningResourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550104/twinningRelations/789",
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Fri, 28 Jun 2013 17:51:59 GMT
```

D.36 Notify a client about Twinning status toggle event (section 6.8.5.2)

Request:

```
POST /twinning/notifications/77777 HTTP/1.1
Accept: application/json
Content-Type: application/json
Host: applicationClient.example.com

{"twinningEventNotification": {
    "callbackData": "12345",
    "eventType": "ToggledOff",
    "link": {
        "href": "http://example.com/exampleAPI/twinning/v1/tel%3A%2B19585550100/subscriptions/sub001",
        "rel": "TwinningSubscription"
    },
    "twinningResourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550100/twinningRelations/1800",
}}
```

Response:

```
HTTP/1.1 204 No Content
Date: Fri, 28 Jun 2013 17:51:59 GMT
```

D.37 Secondary Device’s server requests a new Twinning Activation Code from the Primary Device’s server (section 6.9.5.1)

Request:

```
POST /exampleAPI/twinning/v1/interMnoActivationCode / HTTP/1.1
Content-Type: application/json
Content-Length: nnnn
Accept: application/json
Host: example.com

{"interMnoActivationCode ": {
    "address": "tel:+19585550103",
    "twinningResourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550103/twinningRelations/333"
}}
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnnn
Date: Tue, 03 Feb 2015 21:32:52 GMT

{" interMnoActivationCode ": {
    "address": "tel:+19585550103",
    "twinningResourceURL": "http://exampleAPI/twinning/v1/tel%3A%2B19585550103/twinningRelations/333",
    “activationCode”: {
       "code": "711711",
       "mno": {
           "name": "MNO-A",
           "ncc": "310410"
       }
   }
}}
```

Appendix E. Operations mapping (Informative)

As this specification does not have a baseline specification, this appendix is empty.

Appendix F. Light-weight Resources (Informative)

As this version of the specification does not define any Light-weight Resources, this appendix is empty.

Appendix G. Authorization aspects (Normative)

This appendix specifies how to use the RESTful Twinning Devices API in combination with some authorization frameworks.

G.1 Use with OMA Authorization Framework for Network APIs

The RESTful Twinning Devices API MAY support the authorization framework defined in [Autho4API_10].

A RESTful Twinning Devices API supporting [Autho4API_10]:

- SHALL conform to section D.1 of [REST_NetAPI_Common];
- SHALL conform to this section G.1.

G.1.1 Scope values

G.1.1.1 Definitions

In compliance with [Autho4API_10], an authorization server serving clients requests for getting authorized access to the resources exposed by the RESTful Twinning Devices API:

- SHALL support the scope values defined in the table below;
- MAY support scope values not defined in this specification.

Scope value	Description	For one-time access token
oma_rest_twinning.all_{apiVersion}	Provide access to all defined operations on the resources in this version of the API The {apiVersion} part of this identifier SHALL have the same value as the “apiVersion” URL variable which is defined in section 5.2. This scope value is the union of the other scope values listed in next rows of this table.	No
oma_rest_twinning.primary	Provide access to all defined operations on the resources which need be exposed to the Twinning client application playing the role of the Primary in a given Twinning relationship	No
oma_rest_twinning.secondary	Provide access to all defined operations on the resources which need be exposed to the Twinning client application playing the role of the Secondary in a given Twinning relationship	No
oma_rest_twinning.toggle	Provide access to all defined operations relating to toggle on/off an existing Twinning relationship	No
oma_rest_twinning.userPref	Provide access to operations relating to managing user preference data of an existing Twinning relationship	No
oma_rest_twinning.userPref_name	Provide access to update the Twinning relationship’s name (which is part of user preference data).	No

Scope value	Description	For one-time access token
oma_rest_twinning.subscription	Provide access to all defined operations relating to managing notification subscriptions.	No
oma_rest_twinning.intermno	Provide access to all defined operations on the resources which need be exposed between the two MNO servers participating in an inter-MNO Twinning relationship	No
oma_rest_twinning.intermnoprimary	Provide access to all defined operations on the resources which need be exposed to the Primary Device's MNO server participating in an inter-MNO Twinning relationship.	No
oma_rest_twinning.intermnosecondary	Provide access to all defined operations on the resources which need be exposed to the Secondary Device's MNO servers participating in an inter-MNO Twinning relationship.	No

Table 2: Scope values for RESTful Twinning Devices API

G.1.1.2 Downscoping

In the case where the client requests authorization for "oma_rest_twinning.all_{apiVersion}" scope, the authorization server and/or resource owner MAY restrict the granted scope to some of the following scope values:

- oma_rest_twinning.primary
- oma_rest_twinning.secondary
- oma_rest_twinning.toggle
- oma_rest_twinning.userPref
- oma_rest_twinning.userPref_name
- oma_rest_twinning.subscription
- oma_rest_twinning.intermno
- oma_rest_twinning.intermnoprimary
- oma_rest_twinning.intermnosecondary

G.1.1.3 Mapping with resources and methods

Tables in this section specify how the scope values defined in section G.1.1.1 for the RESTful Twinning Devices API map to the REST resources and methods of this API. In these tables, the root "oma_rest_Twinning." of scope values is omitted for readability reasons.

Resource	**URL** **Base URL:** **http://{serverRoot}/twinning /{apiVersion}/{userId}**	**Section reference**	**HTTP verbs**			
			GET	**PUT**	**POST**	**DELETE**
Twinning relationships	/twinningRelations	6.1	**all_{apiVersion}** or **primary** or **secondary**	n/a	**all_{apiVersion}** or **primary** or **secondary**	n/a
An individual Twinning relationship	/twinningRelations{twinningId}	6.2	**all_{apiVersion}** or **primary** or **secondary** or **intermno**	**n/a**	**all_{apiVersion}** or **intermnoprimary**	**all_{apiVersion}** or **primary** or **secondary** or **intermno**
Twinning status	/twinningRelations{twinningId}/status	6.3	**all_{apiVersion}** or **primary** or **secondary** or **toggle** or **intermno**	**all_{apiVersion}** or **primary** or **secondary** or **toggle** or **intermno**	n/a	n/a
Twinning user preference parameters	/twinningRelations{twinningId}/userPrefrence	6.4	n/a	**all_{apiVersion}** or **primary** or or	n/a	n/a

Resource	**URL** **Base URL:** **http://{serverRoot}/twinning /{apiVersion}/{userId}**	**Section reference**	**HTTP verbs**			
			GET	**PUT**	**POST**	**DELETE**
				userPref or **userPref_name** or **intermnoprimary**		
All subscriptions to Twinning notifications	/subscriptions	6.5	**all_{apiVersion}** or **primary** or **secondary** or **subscription**	n/a	**all_{apiVersion}** or **primary** or **secondary** or **subscription**	n/a
Individual subscription to Twinning notifications	/subscriptions/{subscriptionId}	6.6	**all_{apiVersion}** or **primary** or **secondary** or **subscription**	n/a		**all_{apiVersion}** or **primary** or **secondary** or **subscription**
subscription duration	/subscriptions/{subscriptionId}/duration	6.7	n/a	**all_{apiVersion}** **or** **primary** **or** **secondary** **or**	n/a	n/a

Resource	URL Base URL: http://{serverRoot}/twinning/{apiVersion}/{userId}	Section reference	HTTP verbs			
			GET	PUT	POST	DELETE
				subscription		

Table 3: Required scope values for: Twinning relationship management

Resource	URL Base URL: http://{serverRoot}/twinning/{apiVersion}	Section reference	HTTP verbs			
			GET	PUT	POST	DELETE
Inter-MNO Twinning activation code	/interMnoActivationCode	6.9	n/a	n/a	**all_{apiVersion}** or **intermnosecondary**	n/a

Table 4: Required scope values for: acquiring/generating inter_MNO Twinning activation code

G.1.2 Use of ‘acr:auth’

This section specifies the use of ‘acr:auth’ in place of an end user identifier in a resource URL path.

An ‘acr’ URI of the form ‘acr:auth’, where ‘auth’ is a reserved keyword MAY be used to avoid exposing a real end user identifier in the resource URL path.

A client MAY use ‘acr:auth’ in a resource URL in place of a {userId} when the RESTful Twinning Devices API is used in combination with [Autho4API_10].

In the case the RESTful Twinning Devices API supports [Autho4API_10], the server:

– SHALL accept ‘acr:auth’ as a valid value for the resource URL variable {userId}
– SHALL conform to [REST_NetAPI_Common] section 5.8.1.1 regarding the processing of ‘acr:auth’.