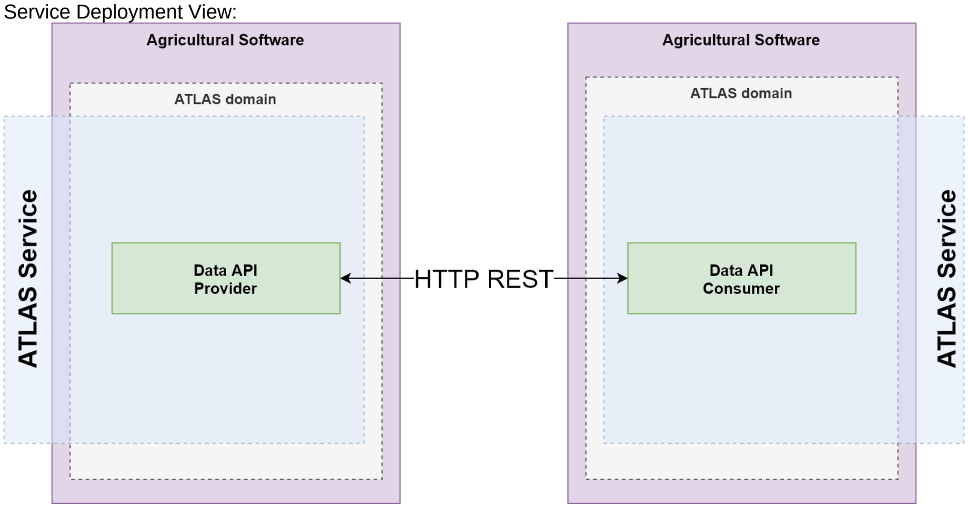
Service Template

Element	Description
ATLAS Service Template ID#	**#**
Name of the ATLAS Service	**Machine Tracking**
Description	The service enables a consuming instance to subscribe to a specific, by the provider offered, set of objects and enables a data transportation the consumer. The data model of the transported information is currently designed for a minimum set of information for tracking of specific machines and car Sepcification. In addition the API provides a REST Endpoint for retrieving data measured in the past.
Version	0.6
Version history and compatibilities	
Preconditions for the implementor	1. Knowledge about REST and HTTPS 2. Knowledge about OpenAPI Specification 3 3. To get the implementation certified: a. minimum requirements of technical data exchange methods (pulling/pushing/subscription) was reviewed and certified b. Technical protocol HTTPS and REST are supported on both sides c. Live telemetry data provider and consumer system can handle (send and/or receive) the pattern described. The consumer system r significant amount of data. d. Implements ATLAS authentication and authorization service
Preconditions for the ATLAS Service	1. Service contracts between the participants (the data platforms and farming software provider) are signed (if required) 2. End user accounts are created in both systems (if required) 3. System paring (authentication and authorization) including registration and approval on both sides is done 4. Currently supported authentication mechanism is oAUTH2 Authorization Code Flow from consumer to provider
Communication endpoint descriptions (API description)	ATLAS-MachineTracking.json

<table>
<tr><th></th><th></th></tr>
<tr>
<td>Sequences and flows</td>
<td>
Sequences for the consumer can be found here.
Sequences for the provider can be found here.

Service Deployment View:

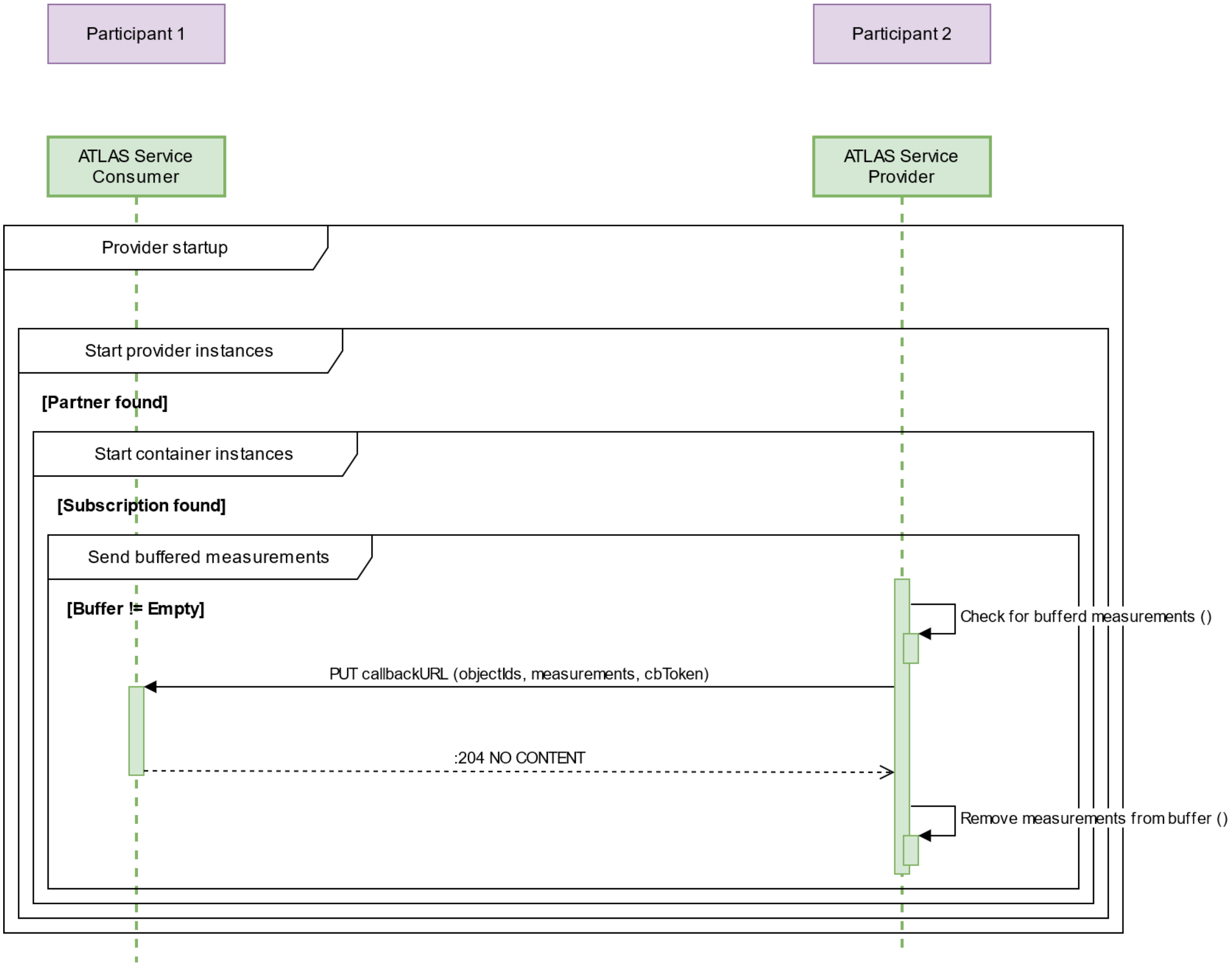


</td>
</tr>
<tr>
<td>Exceptions and error codes</td>
<td>
The error handling is strictly REST. There are no special error codes needed.

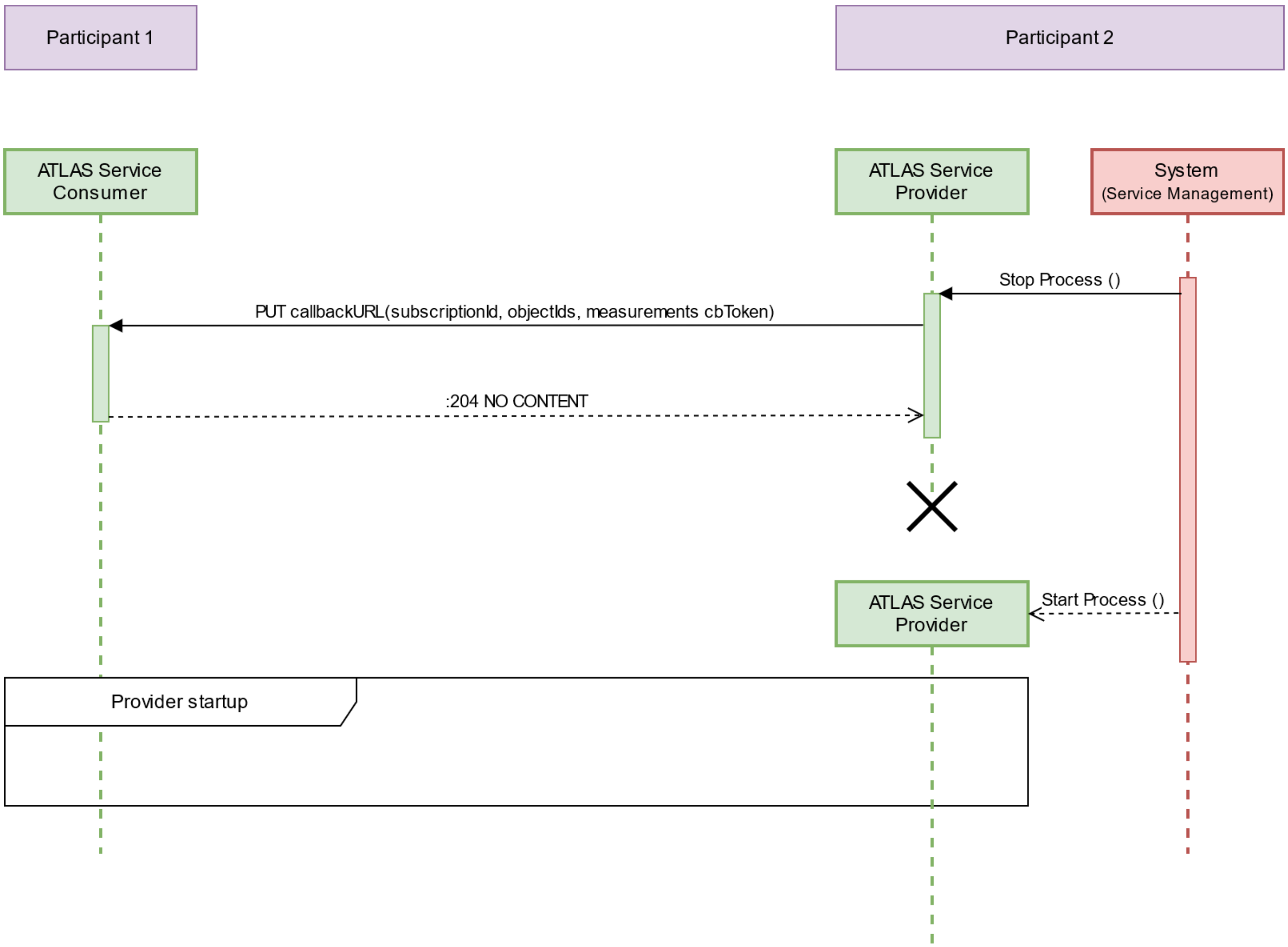
- See ATLAS-MachineTracking.json for error handling
</td>
</tr>
</table>

Provider Sequences

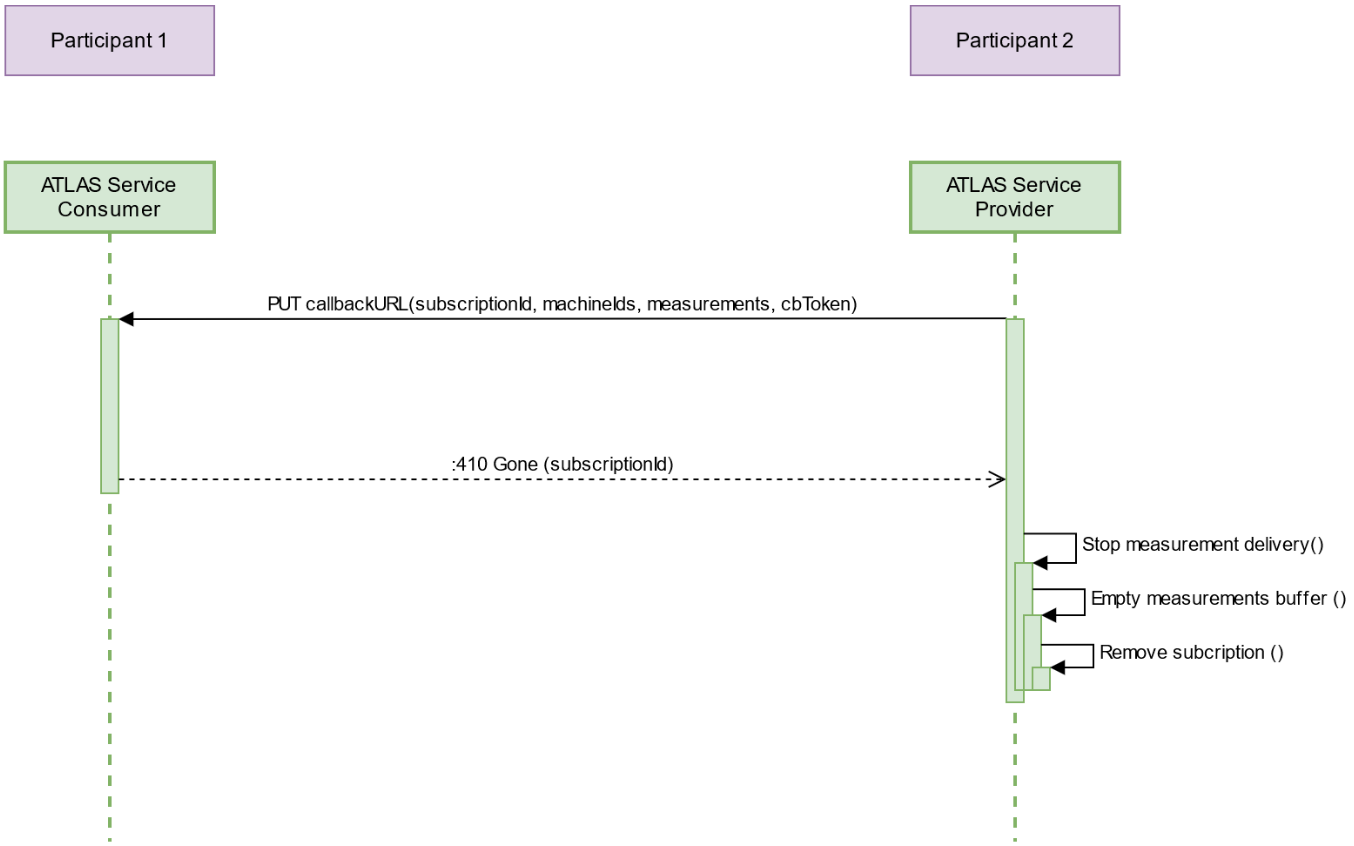
1.1 Provider startup



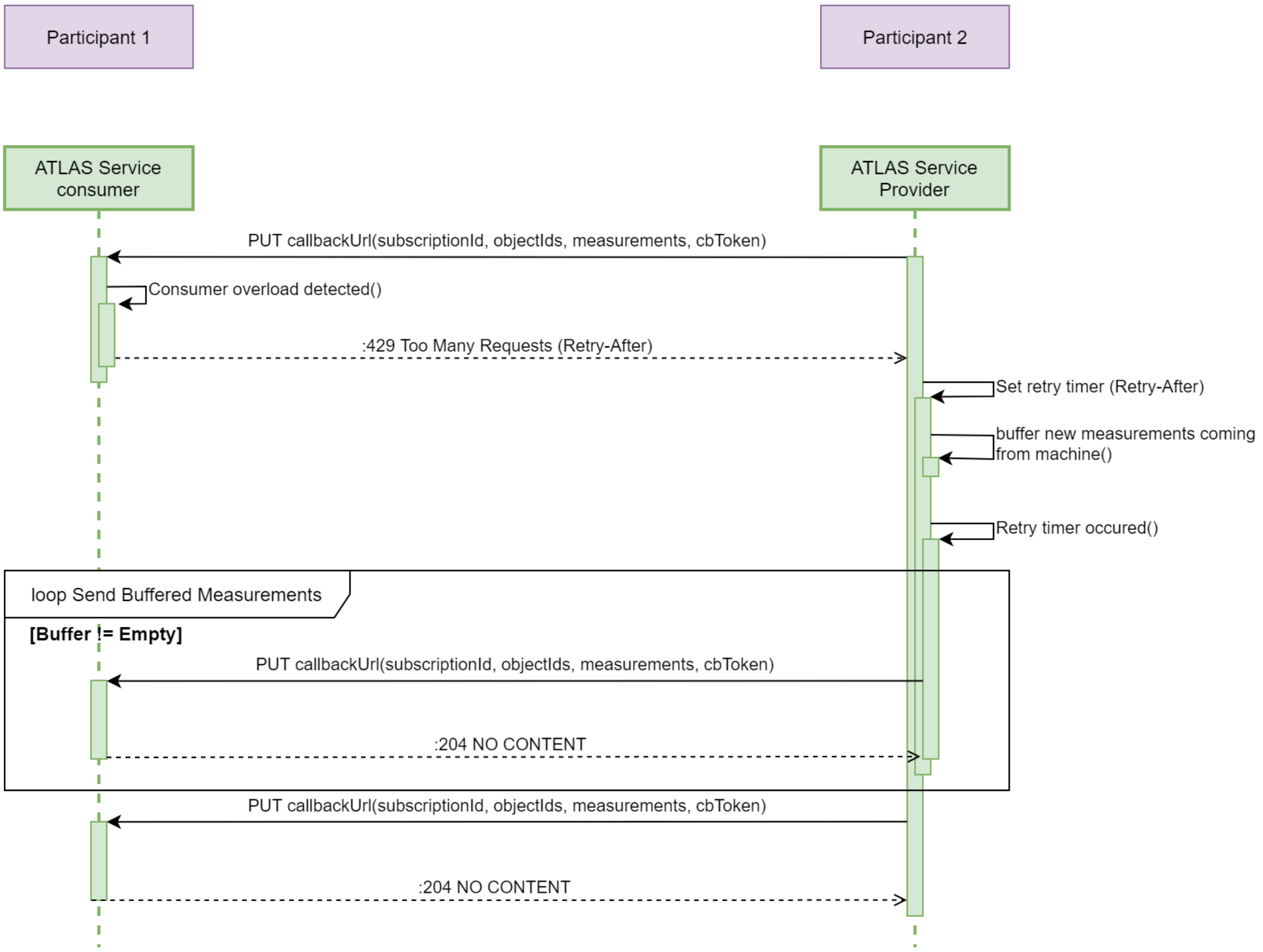
1.2 Provider becomes unavailable



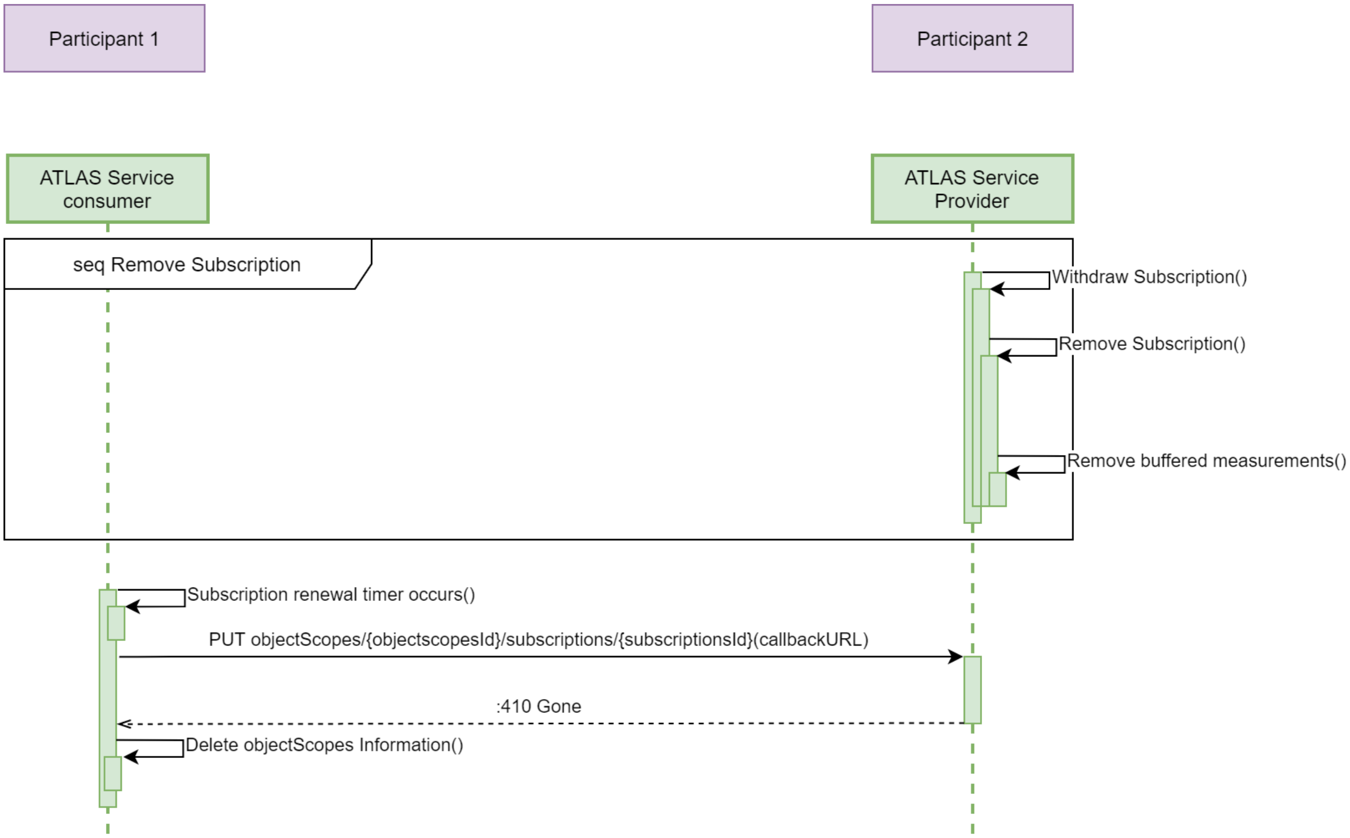
1.3 Provider instance delivers measurements to deleted subscription


Participant 1
Participant 2
ATLAS Service Consumer
ATLAS Service Provider
PUT callbackURL(subscriptionId, machineIds, measurements, cbToken)
:410 Gone (subscriptionId)
Stop measurement delivery()
Empty measurements buffer ()
Remove subcription ()

1.4 Provider Throttling


Participant 1
Participant 2
ATLAS Service consumer
ATLAS Service Provider
PUT callbackUrl(subscriptionId, objectIds, measurements, cbToken)
Consumer overload detected()
:429 Too Many Requests (Retry-After)
Set retry timer (Retry-After)
buffer new measurements coming from machine()
Retry timer occured()
loop Send Buffered Measurements
[Buffer != Empty]
PUT callbackUrl(subscriptionId, objectIds, measurements, cbToken)
:204 NO CONTENT
PUT callbackUrl(subscriptionId, objectIds, measurements, cbToken)
:204 NO CONTENT

1.5 Provider withdraws subscription

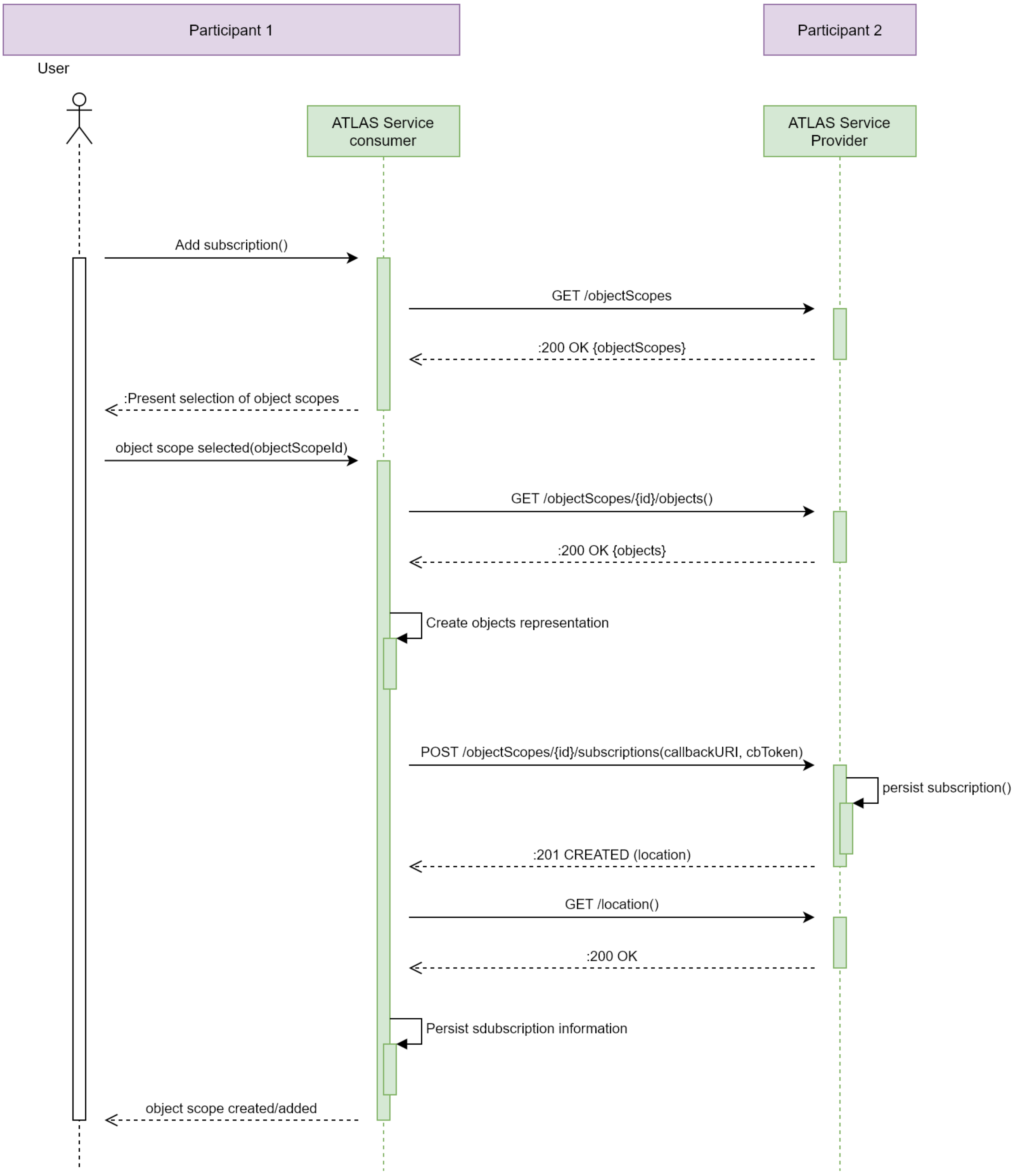

Participant 1
Participant 2
ATLAS Service consumer
ATLAS Service Provider
seq Remove Subscription
Withdraw Subscription()
Remove Subscription()
Remove buffered measurements()
Subscription renewal timer occurs()
PUT objectScopes/{objectscopesId}/subscriptions/{subscriptionsId}(callbackURL)
:410 Gone
Delete objectScopes Information()

Consumer Sequences

1.1 Create new subscription

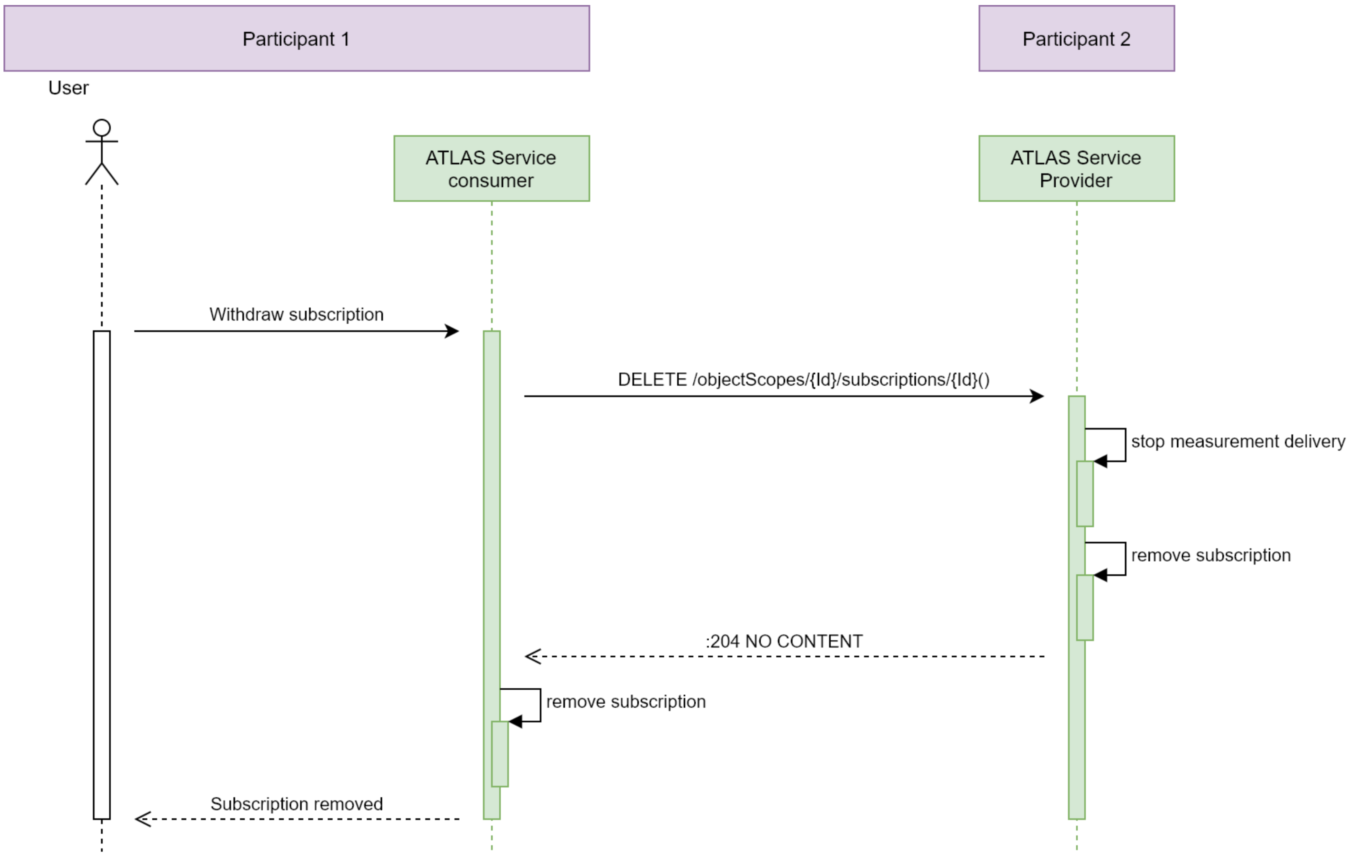
Whenever a data flow shall be established a new subscription needs to be created before.
The subscription, created by the consumer, orders the provider to present data to the configured callback URL.

The sequence shows the needed steps to successfully create a new subscription.


Participant 1
Participant 2
User
ATLAS Service consumer
ATLAS Service Provider
Add subscription()
GET /objectScopes
:200 OK {objectScopes}
:Present selection of object scopes
object scope selected(objectScopeId)
GET /objectScopes/{id}/objects()
:200 OK {objects}
Create objects representation
POST /objectScopes/{id}/subscriptions(callbackURI, cbToken)
persist subscription()
:201 CREATED (location)
GET /location()
:200 OK
Persist sdubscription information
object scope created/added

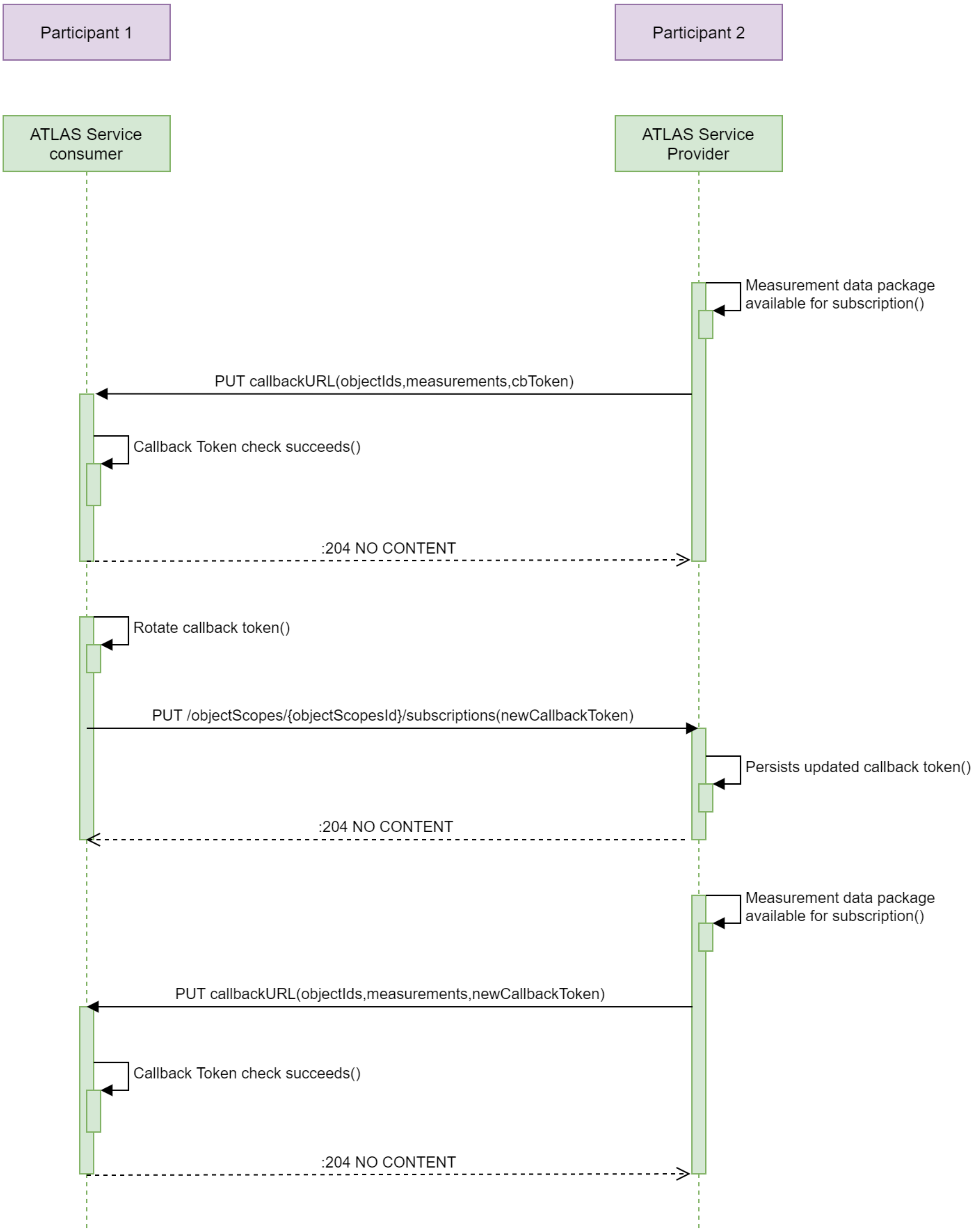
1.2 Consumer removes subscription

Likewise creating a new subscription it is possible to remove one if the data tranportation associated with it is not of interest anymore.



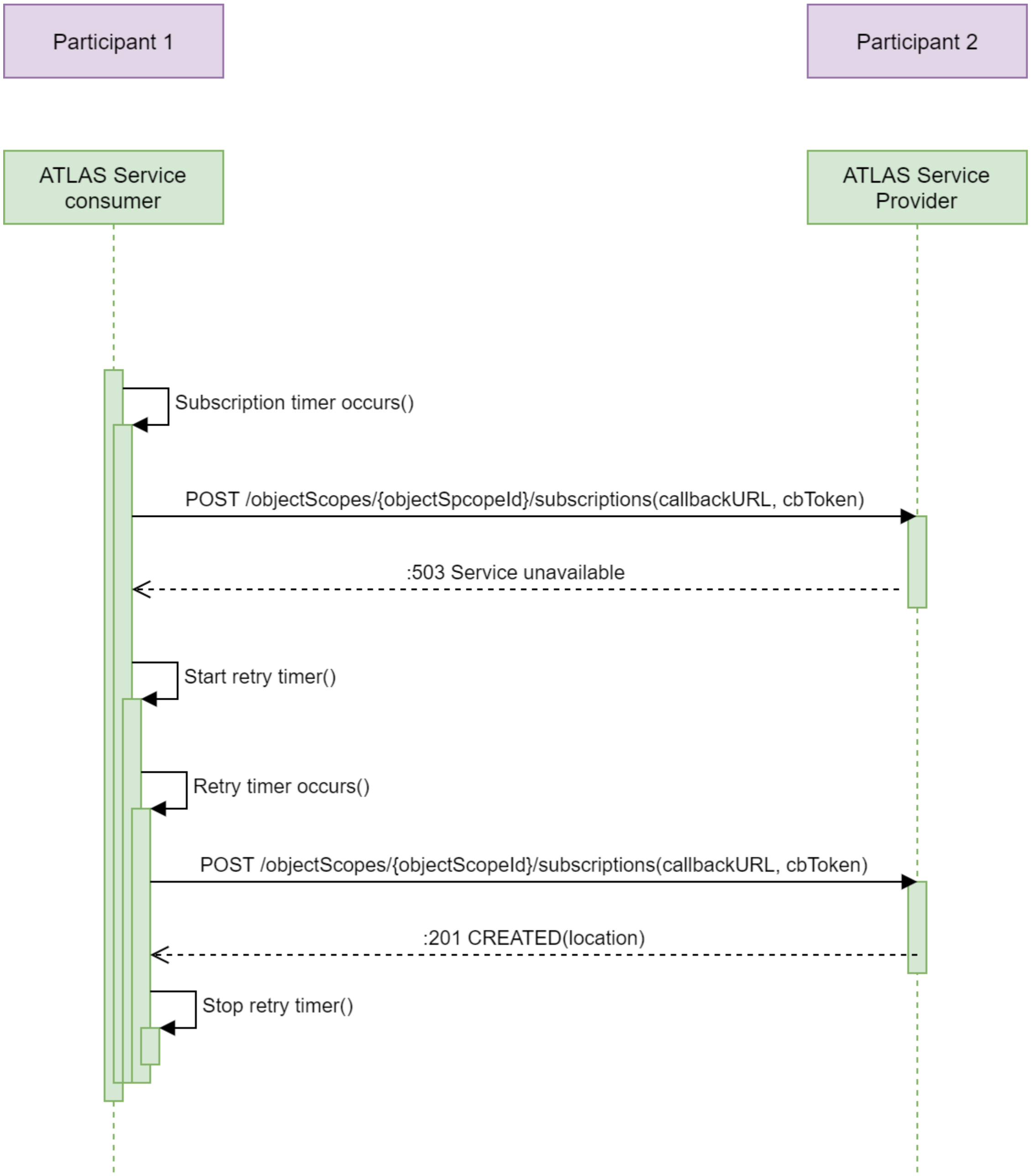
1.3 Callback token update / rotation

Every subscription comes with a callback token to ensure that the data is coming from the desired provider. The provider will always send the token with every call of the callcak URL. The token is then validated before the data is received by the consumer.
To ensure a possibly high level of security it is possible to rotate the token by the consumer like shown in the sequence.


Participant 1
Participant 2
ATLAS Service consumer
ATLAS Service Provider
Measurement data package available for subscription()
PUT callbackURL(objectIds,measurements,cbToken)
Callback Token check succeeds()
:204 NO CONTENT
Rotate callback token()
PUT /objectScopes/{objectScopesId}/subscriptions(newCallbackToken)
Persists updated callback token()
:204 NO CONTENT
Measurement data package available for subscription()
PUT callbackURL(objectIds,measurements,newCallbackToken)
Callback Token check succeeds()
:204 NO CONTENT

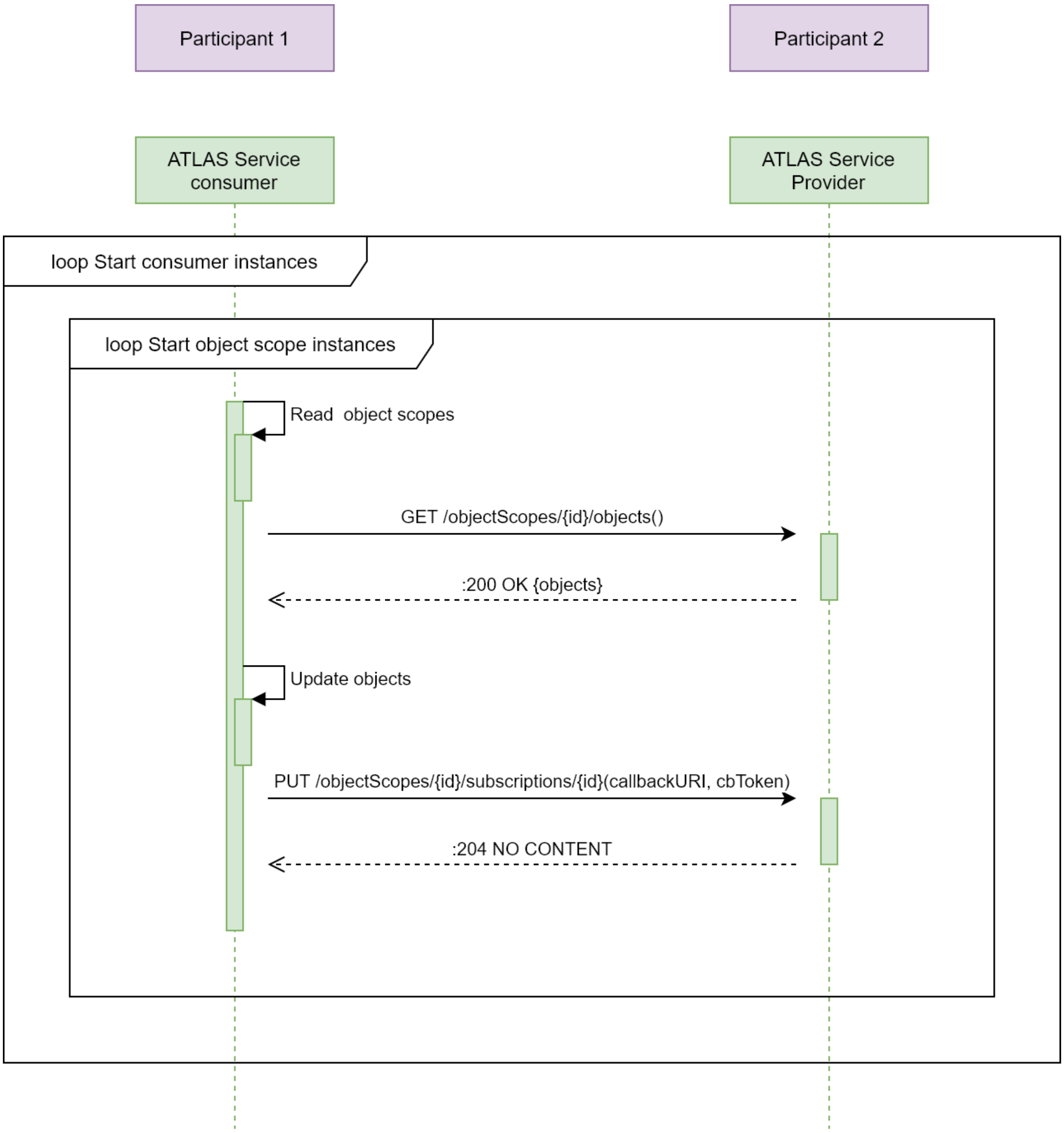
1.4 Subscription fails

REMOVE?



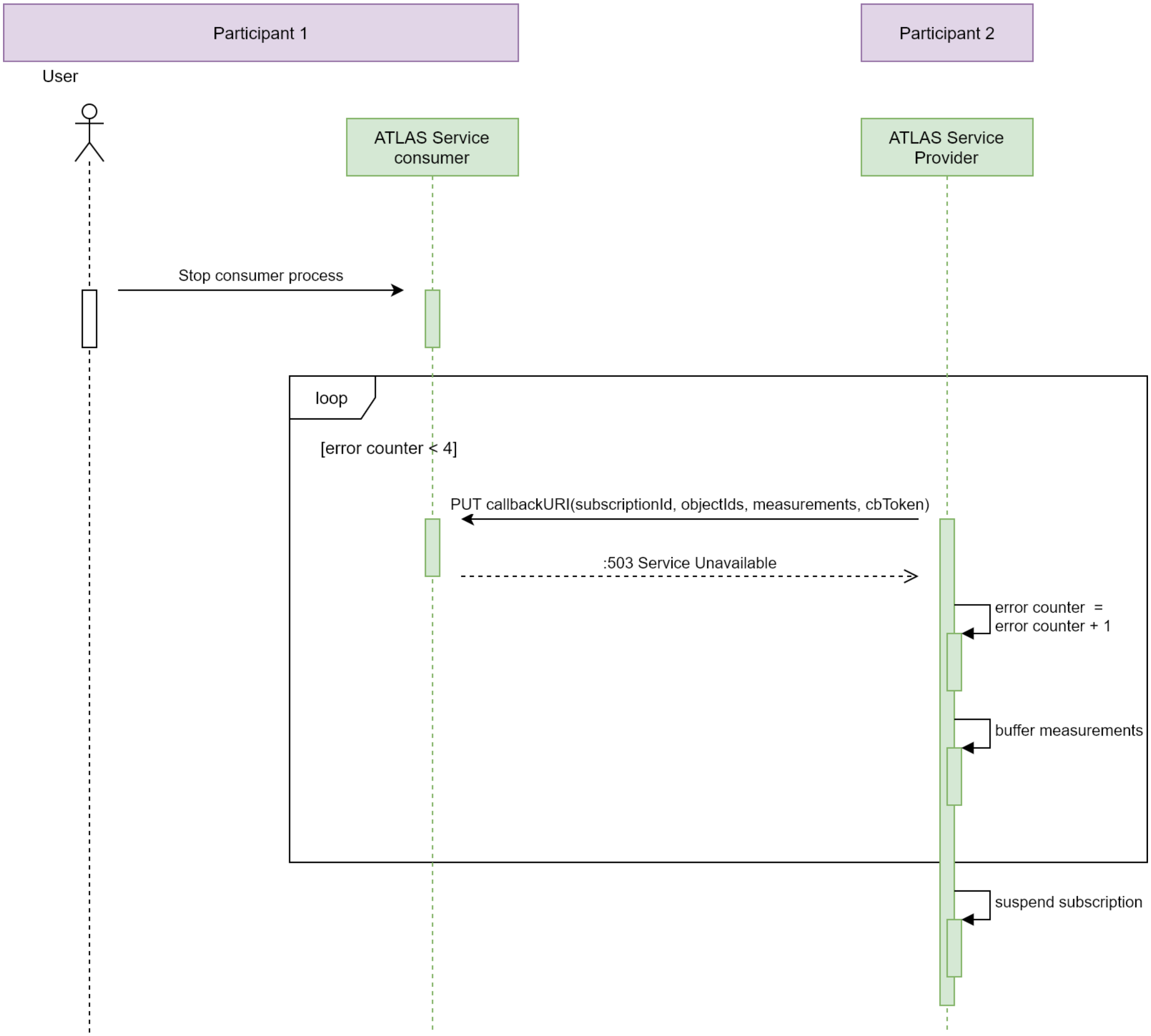
1.5 Consumer Startup

Whenever a consumer has been offline or stopped completely it should rotate through a startup sequence collecting the subscriptions from the provider to ensure that the subscription information on the consumer side are redundant to the provider.
This is, however, only of need if there is a local persistence of the subscriptions in the consumer.


Participant 1
Participant 2
ATLAS Service consumer
ATLAS Service Provider
loop Start consumer instances
loop Start object scope instances
Read object scopes
GET /objectScopes/{id}/objects()
:200 OK {objects}
Update objects
PUT /objectScopes/{id}/subscriptions/{id}(callbackURI, cbToken)
:204 NO CONTENT

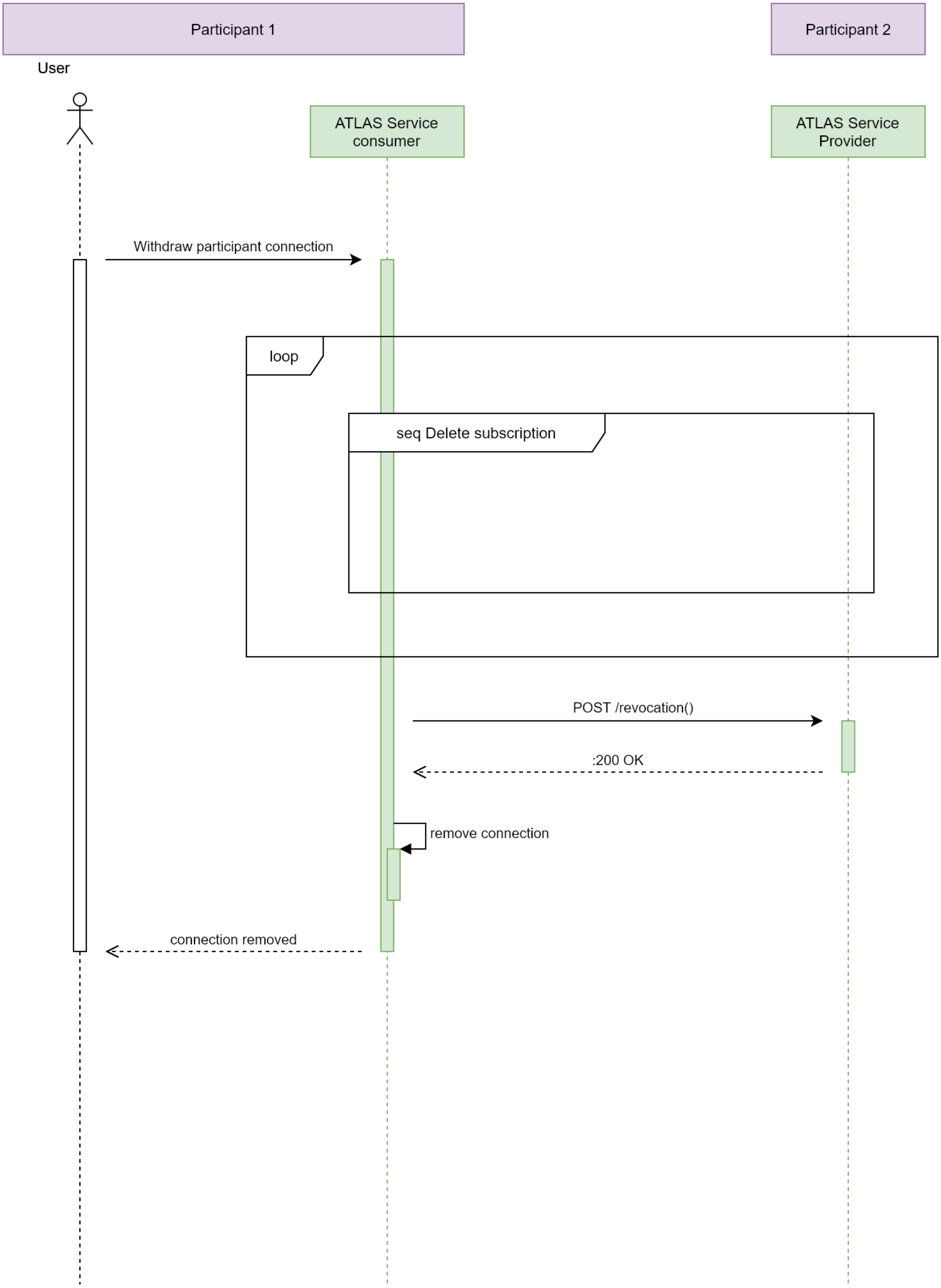
1.6 Consumer becomes unavailable

There will be occasions where a consumer is not able to answer the requests of the callback URLs. This can be a planned maintenance or a severe error state.
If that happens it is assumed that the provider will retry to send the data for three times. If the transmission stil fails the provider will suspend the subscription and wait for the consumer to come back online and once again reactivate the subscriptions. See "Consumer Startup"



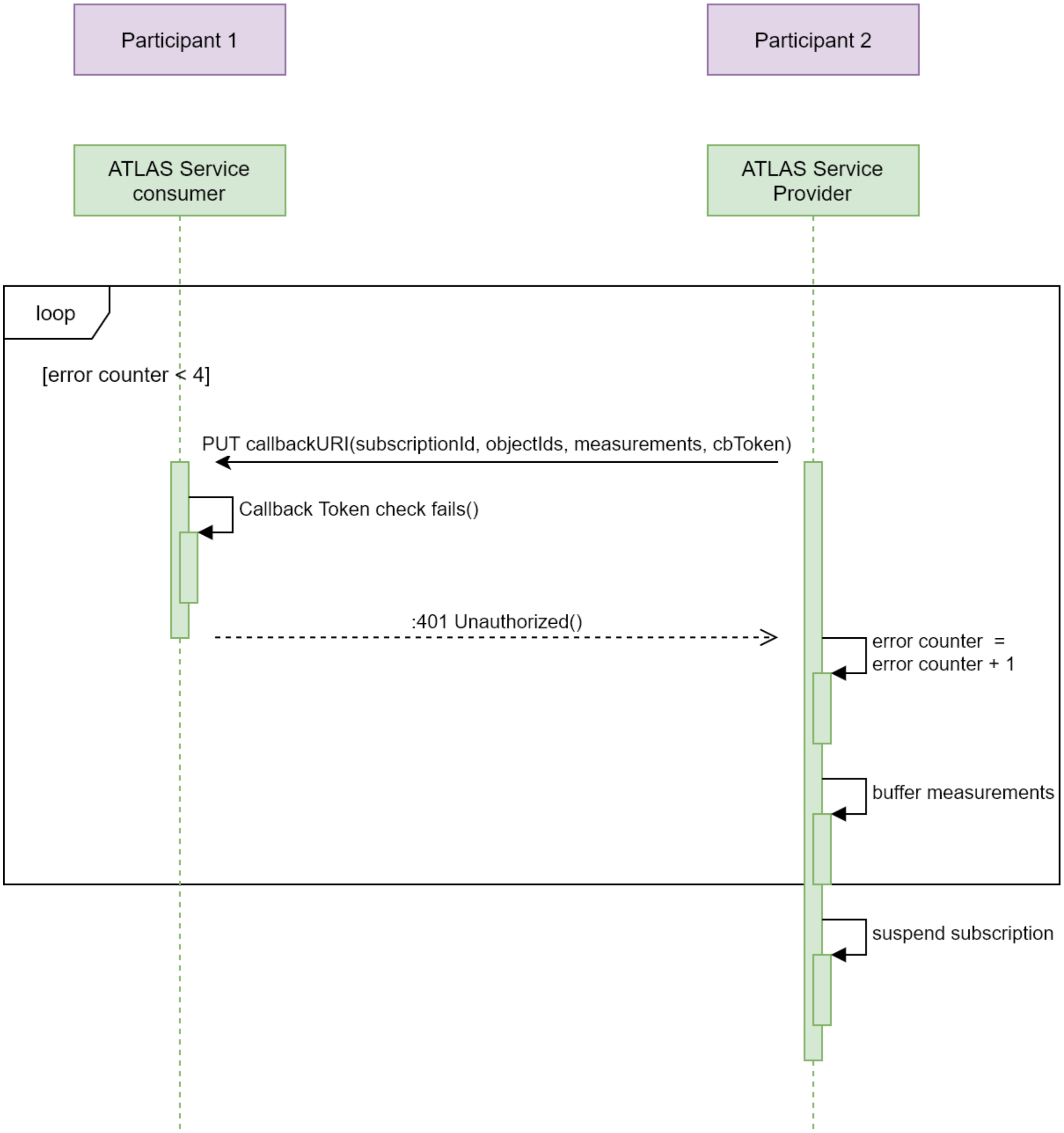
1.7 Consumer withdraws participant connection

For a graceful abortion of data transmission it is assumed that the consumer will delete the subscriptions on the provider side before revoking its access to the server by calling the revocation endpoint.


Participant 1
Participant 2
User
ATLAS Service consumer
ATLAS Service Provider
Withdraw participant connection
loop
seq Delete subscription
POST /revocation()
:200 OK
remove connection
connection removed

1.8 Reject data delivery

Same like being completely unavailable it is possible that the consumer rejects the providers data transmission due to a wrong callback token. The provider should retry to send the data and go into suspension of the subscription after three tries.


Participant 1
Participant 2
ATLAS Service consumer
ATLAS Service Provider
loop
[error counter < 4]
PUT callbackURI(subscriptionId, objectIds, measurements, cbToken)
Callback Token check fails()
:401 Unauthorized()
error counter = error counter + 1
buffer measurements
suspend subscription