

SCR API Specification

Last update: October 10th 2023

Version: 1.2.9

Owner: T-Mining NV

This is the specification of the SCR API that can be used by existing systems to interact with the SCR (Secure Container Release) network. This SCR API is to be installed on your own server, as it incorporates a wallet that stores and uses your key pair to sign transactions on the network and decrypt data. After installation, you just set the authentication method and you can start interacting with the API.

We also have an endpoint in our UAT environment that you can use to do some basic tests on this API. You can find the details of this sandbox API in the last section.

This document contains an overview of all API functionality that is available in the current API or that is coming up in the next release. If functionality is described but not available yet, this is indicated in the text.

Apart from this, this document contains information on notifications and the callback mechanism, authentication and the sandbox API.

Please note that if you are not using the latest version of the API it may not support all functionalities described in this document. Feel free to ask an update to the support team or refer to the last section of this document for the upgrade procedure.

If you have any questions, contact us via support@securecontainerrelease.com.

More information available on <https://www.securecontainerrelease.com/scr-api>.

NOTE: this document provides the specification of the SCR API for Freight Forwarders and Transporters. For more details on the Carrier API or Terminal API, please contact us via sales@t-mining.be.

New in this version

In the newest version of the API and the contracts, the following changes and/or bug fixes are included:

Version 1.2.9

This version includes:

- Changes in the green lights, in line with the new implementation of Nxtport
- Addition of the green lights event notifications
- Addition of the assign barge-endpoints, since these are now available for testing
- Bugfixes for the filters on the GET /releases-endpoint
- Bugfixes for the synchronization of connections in the API
- Bugfixes for the use of circular logfiles
- Bugfixes such that VAT numbers of previous and next parties are returned correctly

You can download the latest version of the API on this page:

<https://www.securecontainerrelease.com/downloads>

Implementation guidelines

The SCR API is a standard REST-API and in that respect, there will be few surprises for your developers. However, to guarantee the future performance and stability of the application, we would like to stress that the guidelines listed below have to be taken into account when interacting with the API.

Always use event notifications instead of polling

Polling the SCR API should be avoided since it will create unnecessary transactions on the blockchain network and therefore can affect the performance of your integration but also of the other parties in the network.

Therefore, you should use the event notification mechanism to receive updates and new releases, and process them in your internal system. This is a very simple mechanism where you register a callback url once and afterwards, all release events will be notified on that callback url. When an event comes in your system can retrieve the details by calling the API, but you should not poll the API for the containers in your transport management system every x minutes.

Field lengths and definitions

All text fields have a default length of 255 characters. Only the terms and conditions field has an unlimited length.

Error handling and time-outs

Since the SCR API connects to other servers in the network, time-outs and other infrastructure related errors are always possible and should be taken into account during development. Retrying a call after a few minutes is the easiest way to deal with this, since most transactions are not time critical.

Simultaneous transactions

The SCR API can accept simultaneous requests without side effects as long as you are retrieving release information. So, when you send a number of simultaneous calls to the GET /releases endpoint, this will not slow down the responses much.

However, calls that change data or retrieving pins are essentially processed sequentially by the SCR API so sending a lot of requests simultaneously will slow down the responses and this should be taken into account at your side. You can send the requests simultaneously and set the time-out high enough, or you can send only a few requests at the same time.

For high volumes of requests we also offer a batch API which can process up to 20 requests at the same time, providing a higher throughput. Please contact us if you want to use these endpoints.

Functional overview

This section lists all endpoints in the API and describes the business logic and data that it provides. We have functionality related to releases, status information, connections and the notification mechanism.

Releases

Get all releases

Get all releases that have been transferred to you. This includes all data except the pincode.

GET /releases

Parameters you can use for **paging**:

- limit: the maximum number of records to return
- page: page number, to be used with limit
- since: timestamp to limit the releases to those modified since then

Parameters you can use for **filtering**:

- bl: the bill of lading number
- container: the container number

These parameters can be used on its own or in combination with others. For example, when using the filter on container number you can also add a filter on bill of lading number, or you can use it on its own. If you specify multiple parameters an AND relationship is used to combine them.

Response:

```
[
  {
    "release_data": {
      "billOfLading": {
        "blNumber": "RELEASE-API-TEST-1612",
        "portOfLoading": "BEANR",
        "portOfDestination": "BEANR",
        "vessel": "AAA",
        "agent": "BBB",
        "stayNumber": "00001",
        "lloydsNumber": "0002",
        "voyageNumber": "0003"
      },
      "container": {
        "containerNumber": "CSQU32345690",
        "containerStatus": "released",
        "pickupLocation": "ECTDCD",
        "turnInLocation": "terminal_locode",
        "turnInReference": "aref",
        "termsAndConditions": "base64 encoded data",
        "validFrom": "2017-01-05T14:00:00.000Z",
        "validUntil": "2025-01-05T14:00:00.000Z",
        "isoTypeCode": "45G1"
      }
    },
    "address": "0x79ED95e778d1C3284636Ab0Be741Ce6836252Fb0",
    "deleted": false,
    "blocked": false,
    "encryptedPin":
"18570bfece4999db5cd9b6102f9cfff472acb5d7e1470bdde8a03cc3db01a6dc1aaf7e132984e
7b2b6d7a07aa021f79c833636d8d9bfe4a62a3bef06567b701c579673768ad435d216050edda2
f2b99709cc542b65c226f0bfd4f30c8500a4eba300cd538bc565409fe01d01d0d0c23d42bab76
c7efe283e89f6e94a5f1b663141aecfe3a97bfeb826996656f622d97e05d4dd7f950f15a9312c
507a90ed58f435b13878adf09d2c727fb518ba8d566bbb8c9a9fb2a1a6fef7a73a55ec6bc05b14
dc6f3cadc201a941f9af018ab1e54f88e0d031e92968a99fa16f9359b8fb943409736f7d2d20e
e0011ed1a817e1a12cd760b9473077b6daa533521563dcacf322d6674a4b4417dd09a42a228211
289bfedcd07529f983cd26387abfe2b1cd4b6817635d788d6d855104055f7ed6102f0604ac438
2204eddfb9e62a9a3c199182319c948d0dabfc62080a93b0c8dac0067431751a4e1e9864fd5d5
48cd93f9c4edb0b95e0edb500f7846ee05dd13d7578f06c26d11ef107dd0713cc1b42f2e70a75
a1a70b5ba107c2eeb6b1aba0723dbf3f8eee674a8ba3ff41da47d9fb91587d9e9c4143f4dad2
c9aa61b444c953720108af392ff3ff85a21542a2100eff453b8c9ffc307200b2f2ce89a2284716
ab14a05b133c22b58359cf9183522fdab46cf033a39c7b262bb7e1015be8c6af837a0a48993a3
1600db207505f3c9eaf1c3e",
    "gateOut": null,
    "createdAt": "2021-12-20T14:17:05.623Z",
    "updatedAt": "2022-02-16T08:52:13.251Z",
    "owner": "0x79F98099b2Bf4e5a47244E0cDbA500F68b567ce5",
    "creator": "0xa2039dC115B5904c224FEc057684fcCcc88E2f05",
    "greenLights": {
      "total": "green",
      "events": {
        "customs": {"color": "green", "value": "released", "updatedAt":
"2023-07-10 13:00"},

```

```

    "pickuplight": {"color": "gray", "value": "unknown", "updatedAt": ""},
    "gateoperation": {"color": "green", "value": "gateoutreceived",
"updatedAt": "2023-07-12 14:16"},
    "terminalready": {"color": "green", "value": "released", "updatedAt":
"2023-07-10 12:00"},
    "customsprogress": {"color": "gray", "value": "unknown", "updatedAt":
""},
    "commercialrelease": {"color": "green", "value": "ok", "updatedAt":
"2023-07-05 13:37"},
    "terminaloperation": {"color": "green", "value": "discharged",
"updatedAt": "2023-07-10 11:50"}
    }, {"source": "NxtPort"}
    "version": "1.2.1",
    "pinRetrieved": false,
    "previousOwner": "0xa2039dC115B5904c224FEc057684fcCcc88E2f05",
    "nextOwner": null,
    "conditions": []
  }
]

```

Field description

The fields illustrated in the output above are explained in the table below in greater detail.

Field	Explanation
blNumber	The bill of lading number
portOfLoading	A text description of the port of loading, supplied by the carrier
portOfDestination	A text description of the port of destination, supplied by the carrier
vessel	A text description of the incoming vessel, supplied by the carrier
agent	A text description of the agent that released the container, supplied by the carrier
stayNumber	The stay number of the incoming vessel, supplied by the carrier
lloydsNumber	The Lloyds number of the incoming vessel, supplied by the carrier
voyageNumber	The voyage number of the incoming vessel, supplied by the carrier
containerNumber	The container number of the container that is released
containerStatus	The base status of the release, can be amended with extra blocks etc (see below)
pickupLocation	A text description of the terminal where the full import container can

	be picked up
pickupLoCode	The UN Locode of the pickup location
turnInLocation	A text description of the terminal or depot where the empty import container has to be returned
turnInLoCode	The UN Locode of the turn in location
turnInReference	The reference that has to be used when returning the empty container
termsAndConditions	A HTML object in which the carrier can include terms and conditions
validFrom	The date and time from which the release is valid on the terminal, note that this date can be before the containers is discharged from the vessel and thus does not indicate availability for pick up
validUntil	The date and time until which the release is valid on the terminal
isoTypeCode	The ISO size and type code of the container
address	The unique address of this release
deleted	A boolean indicating the release was deleted by the carrier, this is a final state
blocked	A boolean indicating the release was temporarily blocked by the carrier, which means the container cannot be picked up at the terminal
encryptedPin	The encrypted pin of the release but should not be used, please use the getPin endpoint to retrieve the pin instead
gateOut	The date and time the container was picked up at the terminal
createdAt	The date and time the release was created by the carrier
updatedAt	The date and time the release was last updated
owner	The address of the owner of the release, but you should use the previous/self/nextOwner fields below instead to identify the owner
creator	The address of the carrier that created the release, see the table below for possible values: <ul style="list-style-type: none"> ● CMA/CGM Belgium: 0x2e13e5323785818Cb8AFC57E66431377712A4b91 ● CMA/CGM Nederland: 0xA82F30D75e4486864Cf65c83e74F8DaB7fc2fEaE

	<ul style="list-style-type: none"> • Hapag-Lloyd Benelux: 0x3041D538A72D73116D6ce13B6cda468d2cECA831 • MSC Belgium: 0xE51e6BBf84e23De4785Edb7096460Ec9956A5B34 • MSC Nederland: 0x9F2377dbb208e233Fb3096539eE5eAA78553c408 • One Nederland: 0xB4F7852d6a3A6537eeCc0465e44C3Dc7112E16D1
version	The version of the smart contract of the release, not to be used
pinRetrieved	A boolean indicating the pin was retrieved
previousOwner	The address of the previous owner, i.e. the company that transferred the release to your company
previousOwnerName	The company name of the previous owner
previousOwnerVat	The VAT of the previous owner
nextOwner	The address of the next owner, i.e. the company that you transferred a release to (if you did)
nextOwnerName	The company name of the next owner
nextOwnerVat	The VAT of the next owner
selfOwner	The address of the company that is calling the API
selfOwnerName	The company name of the company that is calling the API
selfOwnerVat	The VAT of the company that is calling the API
conditions	Not used at the moment
greenLights	Contains status information we receive from for instance a terminal or PCS (see below for details)

Green lights

The green lights implementation can differ by port. The table below shows the information we can provide for the Port of Antwerp, where this information is provided by Nxtport.

Please note the “total”-value is provided by T-Mining for you convenience and uses the following logic (in line with CPU):

- The total light is green when the four relevant lights are also green:
 - Commercial release
 - Terminal discharge
 - Terminal release
 - Customs light
- The total light is yellow when the customs light is yellow
- In all other cases, the total light is red

When the four individual lights become green is documented below.

Green light	Value	Colour	Interpretation
commercialrelease	ok	green	Container has been commercially released.
	nok	red	Container has been commercially blocked
	unknown	gray	No commercial release information received ye
	blocked	red	Container has been commercially blocked
	expired	red	Commercial release has expired
customs	notreleased	red	No release information received yet.
	selectedforscan	yellow	Container has been selected for scanning by competent customs authority. Scanning procedure must be followed.
	released	green	Container has been released by competent customs authority.
	transhipment	yellow	Container can be released under transhipment
	portequalisation	yellow	Container can be released under port equalisation
	favv	yellow	Container can be released under FAVV
	documentarycontrol	red	Container is subject to documentary control
customsprogress	unknown	gray	No information received from customs
	partiallycleared	yellow	At least 1 item of at least 1 BL is fully cleared
gateoperation	unknown	gray	Container has not left the terminal yet according to Terminal Operator.
	gateoutreceived	green	Container has left the terminal according to Terminal Operator.
	gateoutcancelled	gray	Terminal operator has canceled previous gate out message.
	gateincancelled	gray	Terminal operator has canceled previous gate in message.
pickuplight	unknown	gray	No Pick up Right has been generated yet for the

			container.
	assigned	green	Pick up Right has been generated, i.e. a driver has been assigned.
	revoked	red	The assignment of a driver has been revoked, awaiting new assignment.
terminaloperation	unknown	gray	No information on discharge status yet received.
	loaded	red	Container is still on board of vessel.
	discharged	green	Container has been discharged from vessel (import).
	loadcancelled	gray	Terminal operator has canceled previous load message.
	dischargecancelled	gray	Terminal operator has canceled previous discharge message.
	gatein	green	Container has arrived on terminal (transshipment).
terminalready	unknown	gray	No information on release status yet received.
	selectedforscan	yellow	Container has been selected for scan by Terminal Operator. Scanning procedure must be followed.
	blocked	red	Container has been blocked on terminal by Terminal Operator.
	released	green	Container has been released on terminal by Terminal Operator.
	archived	red	Release Right has been archived.

Source:

<https://documentation.nxtport.com/certified-pickup/cpu-variables-overview#CPUvariablesoverview-GreenLighttable>

Get one release

Get the data of a single release identified by its address, excluding the pincode.

GET /releases/{address}

The response is basically the same as above, apart that this endpoint returns one record instead of an array of records.

```

{
  "release_data": {
    "billOfLading": {
      "blNumber": "RELEASE-API-TEST-1612",
      "portOfLoading": "BEANR",
      "portOfDestination": "BEANR",
      "vessel": "AAA",
      "agent": "BBB",
      "stayNumber": "00001",
      "lloydsNumber": "0002",
      "voyageNumber": "0003"
    },
    "container": {
      "containerNumber": "CSQU32345690",
      "containerStatus": "released",
      "pickupLocation": "ECTDCD",
      "turnInLocation": "terminal_locode",
      "turnInReference": "aref",
      "termsAndConditions": "base64 encoded data",
      "validFrom": "2017-01-05T14:00:00.000Z",
      "validUntil": "2025-01-05T14:00:00.000Z",
      "isoTypeCode": "45G1"
    }
  },
  "address": "0x79ED95e778d1C3284636Ab0Be741Ce6836252Fb0",
  "deleted": false,
  "blocked": false,
  "encryptedPin":
"18570bfece4999db5cd9b6102f9cfff472acb5d7e1470bdde8a03cc3db01a6dc1aaf7e132984e
7b2b6d7a07aa021f79c833636d8d9bfe4a62a3bef06567b701c579673768ad435d216050edda2
f2b99709cc542b65c226f0bfd4f30c8500a4eba300cd538bc565409fe01d01d0d0c23d42bab76
c7efe283e89f6e94a5f1b663141aecfe3a97bfeb826996656f622d97e05d4dd7f950f15a9312c
507a90ed58f435b13878adf09d2c727fb518ba8d566bbb8c9a9fb2a1a6fefaf73a55ec6bc05b14
dc6f3cacd201a941f9af018ab1e54f88e0d031e92968a99fa16f9359b8fb943409736f7d2d20e
e0011ed1a817e1a12cd760b9473077b6daa533521563dcacf322d6674a4b4417dd09a42a228211
289bfedcd07529f983cd26387abfe2b1cd4b6817635d788d6d855104055f7ed6102f0604ac438
2204eddfb9e62a9a3c199182319c948d0dabfc62080a93b0c8dac0067431751a4e1e9864fd5d5
48cd93f9c4edb0b95e0edb500f7846ee05dd13d7578f06c26d11ef107dd0713cc1b42f2e70a75
a1a70b5ba107c2eeb6b1aba0723dbf3f8eee674a8ba3fff41da47d9fb91587d9e9c4143f4dad2
c9aa61b444c953720108af392ff3ff85a21542a2100eff453b8c9ffc307200b2f9ce89a2284716
ab14a05b133c22b58359cf9183522fdab46cf033a39c7b262bb7e1015be8c6af837a0a48993a3
1600db207505f3c9ea1f1c3e",
  "gateOut": null,

```

```

"createdAt": "2021-12-20T14:17:05.623Z",
"updatedAt": "2022-02-16T08:52:13.251Z",
"owner": "0x79F98099b2Bf4e5a47244E0cDbA500F68b567ce5",
"creator": "0xa2039dC115B5904c224FEc057684fcCcc88E2f05",
"greenLights": {
  "total": "green",
  "events": {
    "customs": {"color": "green", "value": "released", "updatedAt":
"2023-07-10 13:00"},
    "pickuplight": {"color": "gray", "value": "unknown", "updatedAt": ""},
    "gateoperation": {"color": "green", "value": "gateoutreceived",
"updatedAt": "2023-07-12 14:16"},
    "terminalready": {"color": "green", "value": "released", "updatedAt":
"2023-07-10 12:00"},
    "customsprogress": {"color": "gray", "value": "unknown", "updatedAt":
""},
    "commercialrelease": {"color": "green", "value": "ok", "updatedAt":
"2023-07-05 13:37"},
    "terminaloperation": {"color": "green", "value": "discharged",
"updatedAt": "2023-07-10 11:50"}
  }, "source": "NxtPort"
},
"version": "1.2.1",
"pinRetrieved": false,
"previousOwner": "0xa2039dC115B5904c224FEc057684fcCcc88E2f05",
"nextOwner": null,
"conditions": []
}

```

Get pincode

Get the pincode of the release with the given address.

GET /releases/{address}/pincode

```

{
  "pincode": "12345"
}

```

Transfer a release

Transfer a release to another party. Requires the sender to be the current owner. In case you already fetched the pincode, you can no longer transfer the release. The blockchain account address of the other party must be specified. This must be one of your connections.

PUT /releases/{address}/transfer

Request:

```
{  
  "address": "string"  
}
```

Response: 202 - Request accepted but not yet or partially processed.

Revoke a release

Revoke a release you transferred. The effect is that you become the owner again.

PUT /releases/{address}/revoke

Response: 202 - Request accepted but not yet or partially processed.

Assign a release

Assign a release to a driver or barge. This request allows you to assign a release to a driver or barge. This will be used by the terminal to validate the identity of the driver when he presents himself at the gate, or in the case of a barge when the pre-announcement (copino load) is received.

For pick-up by truck

PUT /releases/{address}/assign

```
{  
  "firstName": "string",  
  "lastName": "string",  
  "email": "string",  
  "phone": "string",  
  "type": "driver",  
  "identification": "string"  
}
```

Response: 204 - Request accepted

For truck drivers, these fields are mandatory in every request:

- firstName
- lastName
- type
- identification

In case of a truck driver, the identification code is in Antwerp the Alfapass number of the driver.

Please note that the API will create and update drivers in the web application automatically. So any driver you assign using the API can also be seen in the web application and used to assign to a release. When the data in your API call is different from the last time this driver was assigned, the data in the web application will be updated too. The key to uniquely identify a driver is the identification number.

For pick-up by barge

For barge, you can use the assign-endpoint and include a type-field, or you can use the more specific assign_barge-endpoint.

First, you can use the general assign-endpoint like this:

```
PUT /releases/{address}/assign
```

```
{
  "type": "barge",
  "name": "Antigoon",
  "identification": "205402090",
  "visitNumber": "BTS2356"
}
```

Response: 204 - Request accepted

Second, you can also use the more specific assign_barge-endpoint. In this case, the type-field can be omitted:

```
PUT /releases/{address}/assign_barge
```

```
{
  "name": "Antigoon",
  "identification": "205402090",
  "visitNumber": "BTS2356"
}
```

Response: 204 - Request accepted

Please note that to update an assignment, you can just do a new call to the endpoints above. You do not need to unassign first.

For barges, the identification is the ENI-number of the barge and the visit number, which is the BTS number in Antwerp.

Therefore, for barges these fields are foreseen:

- name: the name of the barge (mandatory)
- identification: the ENI number of the barge (mandatory)
- visitNumber: the visit number of the barge (optional)

Please note that the visit number (BTS number) is not mandatory. However, the terminal will check this at the moment the loading order (copino) is received. So by that time, the visit number has to be included in SCR. Otherwise, the check of the terminal will fail.

Un-assign a release

Remove the assignment of a release to a driver or barge.

PUT /releases/{address}/unassign

Response: 202 - Request accepted but not yet or partially processed.

Connections

Invite an organization

Use this endpoint to invite an organization to connect to, and exchange releases with. In the body you have to specify:

- A reference: this is a unique reference that is returned when the invited organization has accepted your connection request. This allows you to connect the address of that organization with your internal key (for instance an internal customer number). You can put a client reference number in this field or any other string. This field is not mandatory.
- The name of the organization you want to connect to. This is a mandatory field.
- The VAT number of the organization you want to connect to. This is a mandatory field.
- The contact email of the organization you want to connect to. This email address will be used to send the connection request to that organization. This is a mandatory field.

The governance app will handle the actual invitation, which allows the invited organization to register first if they are not yet on the network, and accept (or refuse) your invitation afterwards.

POST /invitations

```
{
  "reference": "clientref1",
  "name": "Transporter NV",
  "vat": "BE161023354",
  "contactEmail": "info@transporternv.com"
}
```

Get connections

Use this endpoint to get an overview of all connections for your organization. This list contains the name and VAT-number of the connected organizations such that you can match these with your internal database.

The reference you used when you invited the organization is also included (if you did this with the API). So this field can also be used for matching with your internal database, for instance by including your internal supplier reference.

GET /connections

Response:


```
[
  {
    "reference": "9510e738-0bc0-4810-a32a-dc631268779a",
    "name": "Test Entity 2",
    "vat": "ME1115487451",
    "address": "0xa749B9C18362b63f1F6aFc959E776D11A880d962",
    "publicKey": "-----BEGIN PUBLIC
KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAA4cHEu1M2AUVxPStNPMED\n8
B5qGF8fNOu/CX3LJ1aG+z/mMh1w+5xUpd27UpL3hID2xjKsNcsdps6EJxGfQnex\njkRouaJj1dFS
PhPSRpsXul+vxP8w1GnHMnecTRiNrIdigqp/Fvs0SGG0otvCCSEj\n0u/zB/tlvcxlqkCot25T6td
ae79PawpNGKlQPaHqoLbjZswx5JhWt2L4N5mUYUTR\nlflYLTq1MR2OZeGhkb1Op2KMCwUAO740G76
VRowchguXhXhYFS/cqgkAowTP48WAN\nNSF3RO5W9agXdUKvCMnqzLSQXsW0A1C63/eLKdAOfoTCG
0JmwJwtAEmVlARB3vet\n7wIDAQAB\n-----END PUBLIC KEY-----\n"
  },
  {
    "reference": "df84077e-f8d7-4f15-aea0-ce428abe07c2",
    "name": "Test company 10-10-2020",
    "vat": "BE645846454",
    "address": "0x500d82D20d534eB8BA5FBB12f758109944174512",
    "publicKey": "-----BEGIN PUBLIC
KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAYrjYDywx5DgFoBYETxL1\n5
aJ4gqF9eqrQT1En/erEsZ50AkHiKJKUf25QyyvR6ukoG4tFUBHVk5PVH7mXGxxq\nnsaQwx1GBHPzQ
XbVNWRdUzPUfjD4Z874bdHYLEeyijSO+6uSvSDsGbqh/0iVm0UER\nnEz9W18UJLLlXoCkFU1SorhH
H7mGyLOQ/rXFbSvOXaflWEXKN2Pwk32h43jgR+OtG\nnQqjAtN2MOafQ9vTz68aliOXpOD+IM5mrEG
+qpLfX00qkXSciM5CeQsCdHN+uG+T/\nc6rrs+4SKFSV9ORYEpnVOr96JZmpgLuQ7WKA5Te5amUjd
WsD9UVA8oSC41WtExTp\n1wIDAQAB\n-----END PUBLIC KEY-----\n"
  }
]
```

Callback and notifications

Register a callback url for notifications

Subscriptions (or webhooks) serve to specify a callback url that should be invoked when an event occurs.

The following event types are currently supported:

- onRelease
- onAcceptInvitation

Request:

POST /subscriptions

```
{
  "eventType": "string",
  "callback_url": "string"
}
```

The onRelease event type

The onRelease event type covers any of the following:

- a release has been transferred to your organization
- a release that has been transferred to your organization is
 - updated
 - blocked
 - unblocked
 - deleted
 - revoked

Please note that for transfers and revokes of releases that are executed by your own organization (through the API or web app), a notification is also sent to the callback endpoint. This way you can update your internal systems when releases are transferred or revoked manually, in the SCR web application.

When the onRelease event occurs, the callback is invoked with the following data:

- release: the address of the release
- event: a string denoting the event, being one of the following: transfer, update, block, unblock, delete, revoke, greenlights

- organization: the address for which the event is reported
- vat: the vat of the organization for which the event is reported

The last two fields are added for organizations with multiple legal entities. This way, you can collect the notifications using one endpoint and route them to the correct (internal) system using these fields.

Below is an example of a transfer event:

```
{
  "release": "0x24Fc2D465AaAbf47DA7284920A7B5c3b368EC1D9",
  "event": "TRANSFER",
  "organization": "0x78f13e5253a30f0EE079f5DDE281B3F1efe70B94",
  "vat": "BE2020112701"
}
```

and an update of the green lights:

```
{
  "release": "0x24Fc2D465AaAbf47DA7284920A7B5c3b368EC1D9",
  "event": "GREENLIGHTS",
  "organization": "0x78f13e5253a30f0EE079f5DDE281B3F1efe70B94",
  "vat": "BE2020112701"
}
```

After receiving a notification, you need to GET the release in order to receive the latest data. Please note that almost all fields in the release details (from the GET /releases-endpoints) can be updated by the carrier. Please find a list below of these fields:

Field	Can be updated	By	Is updated in practice
billOfLading			
bINumber	yes	carrier	no
portOfLoading	yes	carrier	no
portOfDestination	yes	carrier	no
vessel	yes	carrier	no
agent	no		
stayNumber	yes	carrier	no
lloydsNumber	yes	carrier	no

voyageNumber	yes	carrier	no
container			
containerNumber	yes	carrier	no
containerStatus	yes	SCR app	Yes, when the carrier blocks or unblocks a release this field is updated
pickupLocation	yes	carrier	Yes, when the pickup location (deep sea terminal) changes
turnInLocation	yes	carrier	Yes, when the turn in location (depot) changes
turnInReference	yes	carrier	Yes, when the turn in location (depot) changes
termsAndConditions	yes	carrier	no
validFrom	yes	carrier	no
validUntil	yes	carrier	Yes, when the validity period is updated
isoTypeCode	yes	carrier	no
address	no		
deleted	yes	carrier	no
blocked	yes	carrier	Yes, this can occur when a container has to be scanned or when the carrier blocks the release temporarily
encryptedPin	yes	SCR app	Yes, when the pin is updated, for instance when the release is revoked while the pin was already retrieved
gateOut	yes	terminal	Yes, when the container leaves the terminal
createdAt	no		
updatedAt	yes	SCR app	Yes, when a release is updated
owner	yes	SCR app	Yes, when a release is transferred or revoked
creator	no		
version	no		
pinRetrieved	yes	SCR app	Yes, when the pin is retrieved

previousOwner	yes	SCR app	Yes, when a release is transferred or revoked
nextOwner	yes	SCR app	Yes, when a release is transferred or revoked
conditions	yes	carrier	no
greenLights	yes	Nxtport	Yes

Finally, for the revoke action, the following parties are notified:

- If the API is version 1.2.3 or lower only the current and the new owner receive a notification in the API
- If the API is version 1.2.4 or higher all organizations that are in the ownership chain of the release receive a notification in the API

The onAcceptInvitation event type

The onAcceptInvitation event type serves to get a notification when an invitation has been accepted or refused. When this event occurs, the callback will be invoked with the following data:

- reference: the original reference for the invitation (as set on posting the invitation)
- status: either accepted or refused
- the organization data of the connected organization, as received in the connections endpoint

Error codes

In general the following http return codes are returned:

- 204 - Success
- 401 - Invalid Authorization bearer token
- 500 - Internal server error

When a 500 error is returned, the body of the response contains the error code and description in JSON-format:

```
{  
  "code": 1,  
  "description": "an error"  
}
```

Authentication and configuration of the API

General principles for authentication

The communication between your own systems and the SCR API requires authentication based on signed JWT tokens.

After installation of the SCR API:

- generate a keypair (RSA or ECDSA) for your existing system that will need to communicate with the SCR API
- put this in a file and set the full path of this file in the config file of the SCR API
- export the public key of the wallet (using the wallet CLI) and make sure the resulting file is stored where it is readable by your existing system

Then, for each request from your existing system to the SCR API:

- your existing system should generate a JWT that it signs with its private key, depending on your choice as explained above
- the JWT must have a payload with the following claim:
 - *exp*: expiration time, which should be very short as the JWT should be used only once

The JWT must be put as a bearer token in the authorization header of the request sent to the SCR API.

The SCR API will verify the signature of the JWT using the configured public key in its wallet; if invalid the request is rejected.

For callbacks from the SCR API to your existing system, the same principle applies, but this time the JWT is generated and signed by the SCR API. So your system must get the token from the authorization header, verify the signature with the public key of the SCR API, and check the expiration time.

If you cannot process a bearer token in the headers, you can set the “subscriptionAuthTokenInBody”-setting to “true” in the configuration.json file such that the bearer token is included in the body rather than in the headers of the callback request.

Setting the authentication method of the API

In the *configuration.json* file in the api directory¹, you will find an authentication-setting on line 5 that can be set to your preference:

¹ c:\program files\tmining\wallet\api on Windows and /root/wallet/api on Linux

```
{
  "protocol": "https",
  "baseUrl": "/tmining/api/v1",
  "sqliteDB": "db.sqlite3",
  "authentication": "publicKey.pem",
  ...
}
```

These settings are only for the authentication on the connection between your internal system(s) and the API on your local server. So this traffic does not leave your corporate network and therefore you are free to change this setting.

There are three possible settings:

1. No authentication: remove the line from the configuration file
2. Using a public/private key: put the filename of the pem-file in the line:

```
"authentication": "publicKey.pem",
```

3. Using a secret: put the secret in a file and add the filename to the line:

```
"authentication": "secret.txt",
```

with the file `secret.txt` containing the secret (as text). Be careful! The file should not contain a carriage return or new line character. So, if necessary you can create the file using `echo` on Linux:

```
echo -n 'secret' > secret.txt
```

After changing these settings, the API service should be restarted to apply these changes.

Please take into consideration these guidelines when generating the JWT token:

- The expiration date should be included as an integer, not a string
- The secret has to be base64 encoded in the token
- The token should be encoded with base64URL

Setting the network settings of the API

In the `configuration.json` file in the `api` directory, you will find the network settings on line 2 and 32.

On line 2 you can enable or disable SSL. You can set the protocol to `https` (enabling SSL) or to `http` (disabling SSL):


```
"protocol": "https",
```

When you set https, you also have to specify the SSL certificates to use:

```
"httpServer": {  
  "port": 4431,  
  "ssl": {  
    "serverCert": "certificates/server-cert.pem",  
    "serverKey": "certificates/server-key.pem"  
  }  
}
```

Please use forward slashes in the path.

On line 32, you can update the port that is used to listen to incoming requests:

```
"httpServer": {  
  "port": 5000  
},
```

After changing these settings, the API service should be restarted to apply these changes.

Changing the logging parameters

You can fine tune the logging parameters using the log4js-section in the *configuration.json*-file. The most important setting is the level, which by default is set to “debug”:

```
"level": "debug",
```

You can increase it to “trace” but it will generate a lot of logging.

Apart from this, here is an example for compressing the logs automatically and using rotating logfiles with a limited size:

```
"file": {
  "type": "dateFile",
  "filename": "gateway.log",
  "compress": true,
  "maxLogSize": "100M",
  "backups": 9,
  "layout": {
    "type": "pattern",
    "pattern": "%[%d %p [%f{2}:%l]%" %m%n"
  }
}
```

In this example, a maximum of 10 files are stored on disc. One hot file which is not compressed and will be rotated when the size reaches 100 Mb and 9 compressed files.

Note that you can use all log4js settings, which are documented on the official documentation site:

<https://log4js-node.github.io/log4js-node/file.html>

Creating a token manually

For testing your authentication settings, you can create a JWT token manually and test it with Postman.

You can create a token on for instance <https://jwt.io> with these settings (in this example using a secret):

The screenshot shows the JWT.io interface. At the top, the 'Algorithm' dropdown is set to 'HS256'. The 'Encoded' section shows the resulting token: `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpYXQiOiJlMjOTQwODAxMTgsImV4cCI6MTcyNTcwMjksbnV4OiJkx0H0.rvEtqKn2RSxvr4PzEr6bXMqgNuqaUtprH4TtgaG2_1M`. The 'Decoded' section shows the token's structure:

- HEADER: ALGORITHM & TOKEN TYPE:**

```
{  "alg": "HS256",  "typ": "JWT"}
```
- PAYLOAD: DATA:**

```
{  "iat": 1694080518,  "exp": 1725702918}
```
- VERIFY SIGNATURE:**

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  secret  
)  secret base64 encoded
```

In the payload, the expiration date is key and should be specified as an integer (an epoch). There are no other fields required, so the “issued at” and “expiration time”-fields are sufficient.

This is an example body:

```
{  
  
  "iat": 1694080518,  
  
  "exp": 1725702918  
}
```

with the expiration date on Sep 2024.

Sandbox environment

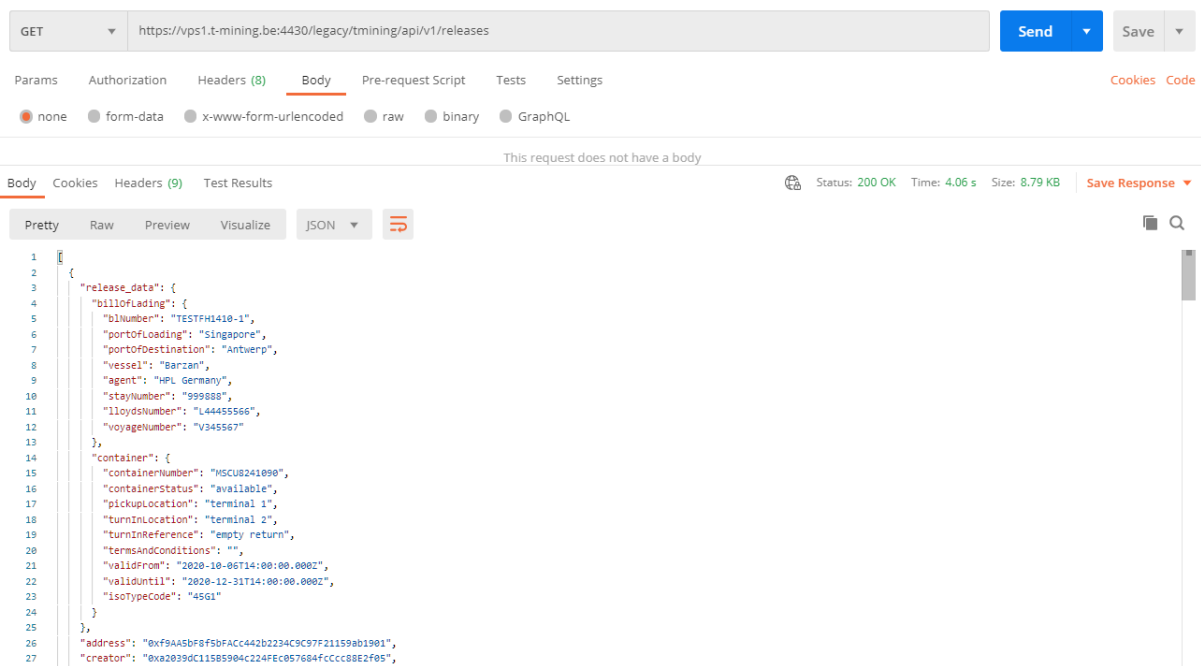
To test the basic operations of the API, you can use the following API endpoint in our UAT environment:

Url: <https://test-admin.securecontainerrelease.com:4430/legacy/tmining/api/v1>

Please note:

- These endpoints do not use authentication, so you can use them without tokens, but in a production environment you should enable authentication
- This also means this data is visible to other companies that are testing in the UAT environment, so please do not add sensitive data

You can retrieve a list of releases using the GET /releases-endpoint and start from there:



```

1
2
3 {
4   "release_data": {
5     "billOfLading": {
6       "blNumber": "TESTFH1418-1",
7       "portOfLoading": "Singapore",
8       "portOfDestination": "Antwerp",
9       "vessel": "Barzan",
10      "agent": "HPL Germany",
11      "stayNumber": "999888",
12      "lloydsNumber": "L44455566",
13      "voyageNumber": "V345567"
14    },
15    "container": {
16      "containerNumber": "MSCU8241090",
17      "containerStatus": "available",
18      "pickupLocation": "terminal 1",
19      "turnInLocation": "terminal 2",
20      "turnInReference": "empty return",
21      "termsAndConditions": "",
22      "validFrom": "2020-10-06T14:00:00.000Z",
23      "validUntil": "2020-12-31T14:00:00.000Z",
24      "isoTypeCode": "45G1"
25    }
26  },
27  "address": "0xf9A45Bf6F5bFACc442b224C9C97F21159ab1901",
28  "creator": "0xa20390C11585904c224Fec057684fccc88E2f05"
  }
  
```

Postman collection

To allow you to test the whole flow we also have a Postman collection that contains endpoints to do all operations on test releases. This can be obtained from support (support@securecontainerrelease.com).

The Postman collection contains a large list of endpoints:

POST Carrier - Create a new release

PUT Carrier - Update a release

PUT Carrier - Transfer a release

PUT Carrier - Revoke a release

PUT Carrier - Block a release

PUT Carrier - Unblock a release

PUT Carrier - Register gate out for a release

GET Get all releases

GET Get one release

GET Get pincode

PUT Transfer a release to test entity 2

PUT Revoke a release

PUT Assign driver to release

PUT Unassign driver

POST Invite a company

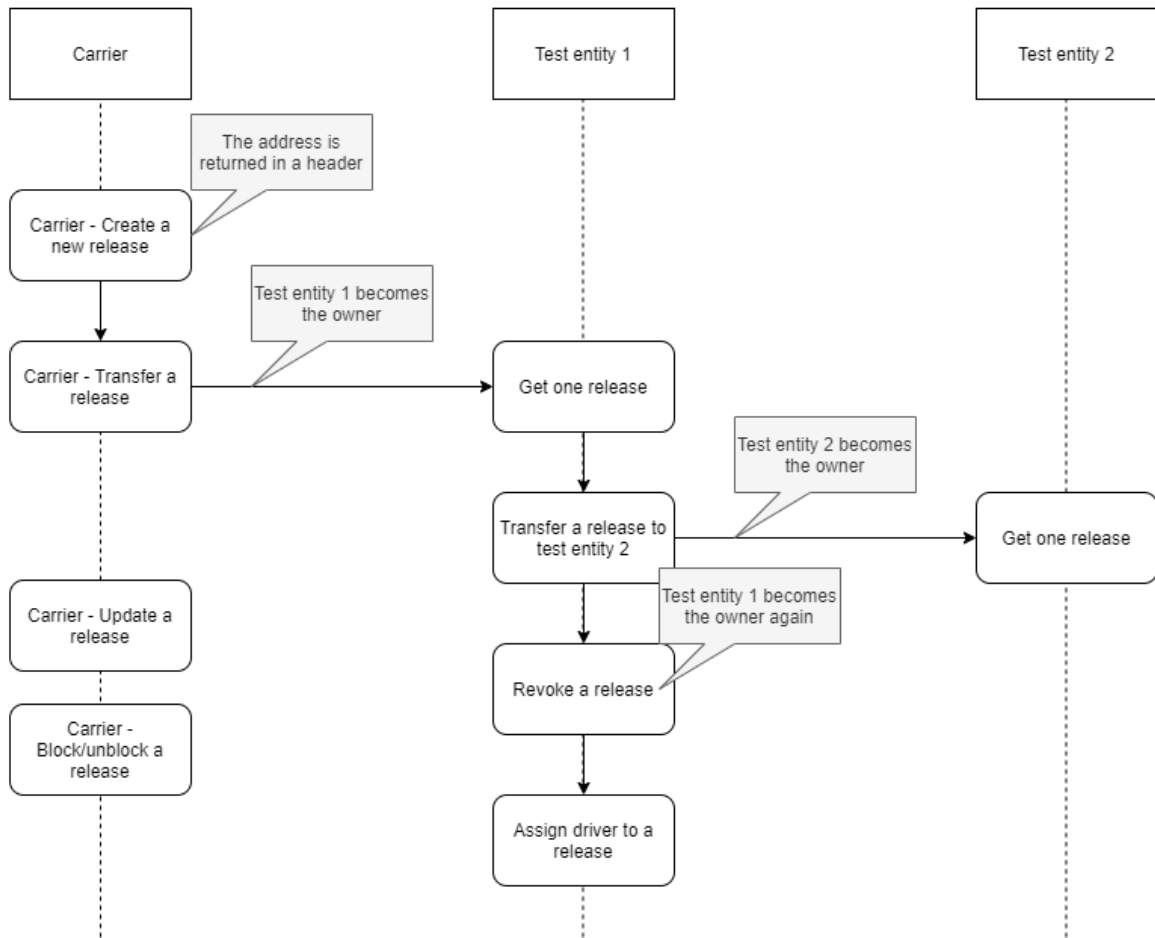
GET Get all connections

POST Update subscription url

It is however important that you follow a basic flow:

- A release is created by the carrier
- By using the “Carrier - Transfer a release” endpoint, you transfer the release to the test organization
- Afterwards the test organization can do a number of transactions using the endpoint without “Carrier” in the description

This is an example of a valid flow:



As you can see, the following parties are available in the collection:

- The carrier to create and update releases
- Test entity 1, which is the subject of this test
- Test entity 2, which is an organization to test the transfer and revoke from test entity 1

Upgrading to the latest version

To upgrade your (production or test) API to the latest version, please use the following procedure:

1. Download the latest version of the AP from the website:
<https://www.securecontainerrelease.com/downloads>
2. Copy the executable file to the correct directory:
 - a. For Windows: `c:\program files\tmining\wallet\api\`
 - b. For Linux: `/root/wallet/api/`
3. Rename the executable file to:
 - a. For Windows: `api.exe`
 - b. For Linux: `api-linux-x64`
4. On Linux, make the file executable: `chmod +x`
5. Restart the API service

When you need assistance, please do not hesitate to contact the support team (support@securecontainerrelease.com).