TransportControl REST API v1

BASE URL

https://[SERVER IP]:[SERVER PORT]/transportcontrol/api/v1

Authentication

Each REST API request is assigned to a registered user, which must authenticate itself using HTTP basic authentication. The credentials are transmitted as Base64 encoded string in the form [USERNAME]:[PASSWORD] inside the authorization header. If authentication fails, error code 401 is returned.

Authorization

Each REST API request requires permission. If permission is not granted, error code 403 is returned. This happens in one of the following cases:

- The administrator settings for the requesting user does not allow resource access in combination with the requested CRUD operation.
- The result of the requested operation would exceed the responsibility limit for this resource type set for the requesting user by the administrator. For example, if a user is allowed to be responsible for up to 10 jobs, this user will not be able to create an 11th job - but deletion requests would be granted.
- The user requests a modification or deletion of a resource that is locked by another user.
- The user requests an operation that is forbidden by the system, e.g., the modification of a read-only resource or resource property.

Request Types

Several request types are available to perform standard CRUD (**C**reate, **R**ead, **U**pdate, **D**elete) operations on TransportControl's resources:

Method	Path	JSON body	Description	Success response
GET	[BASE URL]/[RT]	-	retrieve all resources of type [RT]	200 + resource array
GET	[BASE URL]/[RT]/[ID]	-	retrieve resource	200 + resource
POST	[BASE URL]/[RT]	resource	create new resource; ID will be chosen by server	201 + ID of the newly created resource inside location header
PUT	[BASE URL]/[RT]	resource array	overwrite all resources of type [RT]	204
PUT	[BASE URL]/[RT]/[ID]	resource	create or overwrite resource with a specified ID	204 / 201
PUT	[BASE URL]/[RT]/[ID]/[KEY]	new value as string	set value of a property of a resource	204
DELETE	[BASE URL]/[RT]/[ID]	-	delete a resource	204

For some resource types not all request types are available. If a request type is not available, the server responds with code 404. See the following table for details:

Resource type (RT) ⟶ ↓ Request (method + path)	Contour	Event	Job	Model	Script	Segment	Setting	Target	User	Variable
GET [BASE URL]/[RT]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
GET [BASE URL]/[RT]/[ID]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
POST [BASE URL]/[RT]	✓	404	✓	404	404	✓	404	404	404	404
PUT [BASE URL]/[RT]	✓	✓	✓	404	404	✓	404	404	404	✓
PUT [BASE URL]/[RT]/[ID]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PUT [BASE URL]/[RT]/[ID]/[KEY]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DELETE [BASE URL]/[RT]/[ID]	✓	✓	✓	✓	✓	✓	404	✓	✓	✓

Response Codes

Code	Name	Description
200	OK	request successfully processed; resource as JSON object returned in response body
201	CREATED	resource created; no response body; ID of the newly created resource inside location header
204	NO_CONTENT	request successfully processed; no response body
207	MULTI_STATUS	response to a batch request; individual feedback inside the response body as JSON array
400	BAD_REQUEST	unprocessable request, e.g. resource JSON can't be parsed
401	UNAUTHORIZED	unknown user (wrong username or password)
403	FORBIDDEN	resource access not permitted
404	NOT_FOUND	resource or method not found
422	UNPROCESSABLE_ENTITY	resource in the request body can't be initialized, e.g. invalid ID

Resource Types

Contour	**Description**	definition of building structures, objects from DXF import
	Path	[BASE URL]/contours
	JSON example
```
{
  "id": 788,
  "layer": 0,
  "responsibility": "EXTERNAL",
  "label": "",
  "area": 7.29,
  "center":  { "x": 192351, "y": 474655 },
  "points": [
    { "x": 192222, "y": 476559 },
    { "x": 190446, "y": 474525 },
    { "x": 192480, "y": 472750 },
    { "x": 194256, "y": 474784 }
  ],
  "color": "rgba(0,0,0,0.25)",
  "labelColor": "rgba(0,0,0,0)",
  "labelOffset": { "x": 0, "y": 0 }
}
```
 |
Event	**Description**	definition of time-based tasks, shift system
	Path	[BASE URL]/events
	JSON example
```
{
  "id": "roadworks",
  "isNow": false,
  "type": "SINGLE",
  "start": 1530768600000,
  "stop": 1530892800000,
  "taskList": [
    {
      "trigger": "START",
      "method": "PUT",
      "path": "segments/77/closure",
      "body": "true"
    },
    {
      "trigger": "STOP",
      "method": "PUT",
      "path": "segments/77/closure",
      "body": "false"
    }
  ],
  "responsibility": "DemoUser",
  "properties": {}
}
```
 |
Job	**Description**	transport orders
	Path	[BASE URL]/jobs
	JSON example
```
{
  "targetKey": "TCV",
  "priority": 10,
  "endEvent": "ARCHIVE",
  "appointmentMode": "FIXED",
  "appointmentTime": 1528799778317,
  "appointmentIndex": 1,
  "orderList": [
    {
      "segment": 2,
      "actionCode": 0,
      "actionValue": 0
    },
```
```
// target filter; "TCV" up to full target ID
// 1-100; smaller number = higher priority
// ARCHIVE, REMOVE or RESTART
// ASAP or FIXED
// 0 for ASAP or UNIX time for FIXED
// 0-based index of appointment order
// list of orders; at least 1 order required

// first order: ID of destination segment
// first order: target type dependent action code
// first order: action code dependent value
```
 |

<table>
<tr><td></td><td></td><td>

```
    {
      "segment": 14,
      "actionCode": 0,
      "actionValue": 0
    }
  ],
  "orderIndex": 2,
  "deviation": 6144740,
  "deviationRatio": 0.15,
  "progress": 57,
  "startTime": 1528805879672,
  "status": "COMPLETED",
  "id": "1528377600379",
  "responsibility": "DemoUser",
  "properties": {
    "resetRequest": "false"
  }
}
```

</td><td>

```
// second order: ID of destination segment
// second order: target type dependent action code
// second order: action code dependent value


// [READONLY] index of currently processed order
// [READONLY] deviation relative to appointmentTime
// [READONLY] status DELAYED if greater than 0.25
// [READONLY] 0-100 in percent; for all orders combined
// [READONLY] start of job in UNIX time
// [READONLY] SCHEDULED > ASSIGNED > COMPLETED
// [READONLY] creation timestamp as unique resource ID
// [READONLY] last editor or creator of this job
// can be filled with custom entries
// set to "true" to reset/restart job
```

</td></tr>
<tr><td>Model</td><td>Description</td><td colspan="2">target descriptions</td></tr>
<tr><td></td><td>Path</td><td colspan="2">[BASE URL]/models</td></tr>
<tr><td></td><td>JSON example</td><td>

```
{
  "id": "KATE_short",
  "targetKey": "TCVKS",
  "inputScript": "legacyInput",
  "outputScript": "legacyOutput",
  "shape": "\u003cpath [...]",
  "size": { "x": 770,  "y": 450 },
  "offsetOdometry": { "x": 0,    "y": 0 },
  "offsetProgress": { "x": 250,  "y": 0 },
  "offsetPivot":    { "x": -400, "y": 0 },
  "minRangeOfSight": 300,
  "maxRangeOfSight": 1500,
  "actions": {
    "Charge": {
      "code": 1,
      "minValue": 0,
      "maxValue": 100
    },
    "Lift": {
      "code": 10,
      "minValue": -100,
      "maxValue": 100
    }
  },
  "status": {
    "E-Stop": {
      "index": 66,
      "length": 1,
      "alert": 1,
      "color": "red"
    },
    "Energy [%]": {
      "index": 88,
      "length": 8,
      "alert": 10,
      "color": "auto"
    },
  "errors": {
    "CAN timeout": {
      "index": 111,
      "alert": -1
    },
    "Charge error": {
      "index": 152,
      "alert": -1
    }
  "origin": "SERVER",
  "properties": {
    "maxChunkSize": "640",
```

</td><td></td></tr>
</table>

		<pre>"maxSegConAngle": "30",
		"useSegPosition": "true",
		"useOdoPosition": "true",
		"allowRemote": "true",
		"avoidTraffic": "false",
		"maxOfflinePeriod": "5000",
		"allowUpdate": "true",
		"coronaRadius": "650",
		"maxSpeed": "2"
		}
		}</pre>	

Script	**Description**	definition of plant-specific logistical processes (battery management, traffic control)	
	Path	[BASE URL]/scripts	

	JSON example	<pre>{
		"id": "loadBattery",
		"executionTime": 1530780230009,
		"code": "[...]",
		"log": "",
		"load": 5,
		"responsibility": "GSF_SB",
		"startup": false,
		"errorMessage": "",
		"interval": 1000,
		"errorLine": 0,
		"type": "SERVER",
		"properties": {}
		}</pre>	

Segment	**Description**	definition of route sections	
	Path	[BASE URL]/segments	

	JSON example	<pre>{
		"id": "135",
		"label": "PORT_777",
		"length": 1622,
		"controlPoints": [{
		"x": 321000,
		"y": 503000,
		"a": 270.0,
		"h": 270.0
		},
		{
		"x": 321000,
		"y": 502336
		},
		{
		"x": 320664,
		"y": 502000
		},
		{
		"x": 320000,
		"y": 502000,
		"a": 180.0,
		"h": 180.0
		}
		],
		"heading": "FORWARD",
		"width": 150,
		"color": "rgba(0,0,0,0.31)",
		"labelColor": "rgba(255,81,0,0.75)",
		"labelOffset": {
		"x": 0,
		"y": 0
		},
		"origin": "DemoUser",
		"properties": {
		"condition": "75",</pre>	

		<pre>"spotTurn": "false",
		}
		}</pre>

Setting	**Description**	global system settings
	Path	[BASE URL]/settings

	JSON example	<pre>{
		"id": "udpReceivePort",
		"hidden": false,
		"value": "1111"
		"properties": {},
		"responsibility": "SERVER"
		}</pre>

Target	**Description**	UDP clients / connected PLCs (vehicles, traffic lights, gates...)
	Path	[BASE URL]/targets

	JSON example	<pre>[
		{
		"id": "TCVP60_1",
		"ip": "127.0.0.1",
		"port": 50932,
		"isOnline": true,
		"isValidated": true,
		"isSimulated": true,
		"hasAlert": false,
		"firstTelegramTime": 1530780230395,
		"lastTelegramTime": 1530780398652,
		"lastJobTime": 0,
		"mileage": 874.41,
		"tunedPos": {
		"x": 325000,
		"y": 26950,
		"h": 270.0
		},
		"remoteId": "FREE",
		"rawInput": [
		84,67,86,80,54,[...]
		],
		"rawOutput": [
		84,67,83,69,82,[...]
		],
		"segId": 14,
		"speed": 7,
		"command": 0,
		"route": [],
		"relativeSpeed": 0,
		"steering": 0,
		"seqDataIndex": -1,
		"updateIndex": -1,
		"updateLength": 0,
		"responsibility": "DemoUser",
		"properties": {
		"stop": "false"
		}
		}
		]</pre>

User	**Description**	REST API clients (WMS, browser UI, IoT devices...)
	Path	[BASE URL]/users

	JSON example	<pre>{
		"id": "Eisenbert",
		"access": {
		"models": [...],
		"settings": [...],</pre>

<table>
<tr><td></td><td></td><td><pre>
        "variables": [...],
        "jobs": [...],
        "scripts": [...],
        "contours": [...],
        "targets": [...],
        "events": [...],
        "users": [...],
        "segments": [...]
    },
    "responsibilityCount": {
        "settings": 2,
        "jobs": 5,
        "users": 2,
        "variables": 1,
        "scripts": 8,
        "models": 1,
        "targets": 1,
        "contours": 4
    },
    "responsibilityLimit": {
        "models": 50,
        "settings": 10,
        "variables": 100,
        "jobs": 100,
        "scripts": 100,
        "contours": 1000,
        "targets": 100,
        "events": 100,
        "users": 100,
        "segments": 1000
  },
  "authenticationMethod": "INTERNAL",
  "subscriptions": [],
  "lastRequestTime": 1576251156624,
  "ip": "0:0:0:0:0:0:0:1",
  "properties": {
    "showTrafficData": "false",
    "lock": "true"
  },
  "responsibility": "SERVER",
  "initTime": 1576251005173,
  "userModificationTime": 0,
  "serverModificationTime": 1576251156624
}
</pre></td><td></td></tr>
<tr><td rowspan="3">Variable</td><td>Description</td><td colspan="2">general purpose data storage for script usage</td></tr>
<tr><td>Path</td><td colspan="2">[BASE URL]/variables</td></tr>
<tr><td>JSON example</td><td><pre>
{
  "id": "loadThreshold",
  "value": "50",
  "responsibility": "Eisenbert",
  "properties": {
    "key1": "value1",
    "key2": "value2",
    "key3": "value3"
  }
}
</pre></td><td></td></tr>
</table>

Batch Service

In order to minimize the overhead generated by HTTP requests, several individual requests can be combined into a single request. The following table shows the details:

	Structure	Example
Request method	POST	
Request path	[BASE URL]/batch	

Request body	[{
	"method": [METHOD_1],
	"path": [RELATIVE_PATH_1],
	"body": [BODY_1]
	},
	{
	"method": [METHOD_2],
	"path": [RELATIVE_PATH_2],
	"body": [BODY_2]
	},
	{
	"method": [METHOD_N],
	"path": [RELATIVE_PATH_N],
	"body": [BODY_N]
	}]
		[{
		"method": "DELETE",
		"path": "segments/77",
		"body": ""
		},
		{
		"method": "PUT",
		"path": "targets/TCVP60_7/stop",
		"body": "true"
		},
		{
		"method": "GET",
		"path": "settings/logLevel",
		"body": ""
		}]

Response code	207	
Response body	[RESPONSE_1, RESPONSE_2, RESPONSE_N]	[204, 204, {"id":"logLevel","value":"FINE"}]

Subscription Service

The TransportControl REST server can only react to requests. For clients who want to be automatically informed about changes to certain resources - e.g., vehicle movement or job progress and completion – a subscription service via WebSocket is available. To subscribe please follow these steps:

1. Subscribe to the desired resources via REST API request
   Method: POST; Path: [BASE URL]/subscriptions; JSON body: [RT1, RT2, RT3, ...]
   TransportControl creates the authentication token required for step 2 and transmits this in the location header of the response.

2. Open a WebSocket connection with the following URL
   wss://[SERVER IP]:[SERVER PORT]/transportcontrol/[token]

3. [OPTIONAL] Subsequent change of subscription
   Method: PUT; Path: [BASE URL]/subscriptions/[token]; JSON body: [RT1, RT2, RT3, ...]

After a successful connection, updates of the subscribed resources are sent to the client in time with the core loop (default every 100 ms). The first update contains the complete lists, all other updates contain only the changes to the previous packet. A full subscription would result in a typical update packet like this:

Subscription	Update packet from server	Description

[
  ·
  "targets",
  ·
  ·
  ·
  ·
  ·
  ·
  "jobs",
  "events",
  "variables",
  "scripts",
  ·
  ·
  ·
  ·
  ·
  ·
  ·
  ·
  ·
  ·
  "segments",
  "contours",
  "models",
  "users",
  "settings"
]
	[
	  [1576252516040, 0, 101],
	  [{
	  "id":"TCVSIM01",
	  "lastTelegramTime":1.576252515839E12
	  },
	  {
	  "id":"TCVSIM02"
	  }],
	  null,
	  null,
	  null,
	  [{
	  "id":"testScript"
	  },
	  {
	  "id":"legacyInput",
	  "executionTime":1.576252515939E12
	  },
	  {
	  "id":"legacyOutput",
	  "executionTime":1.576252515939E12
	  }],
	  null,
	  [],
	  null,
	  null,
	  null
	]
		// time sync array: [serverTime, workTime, serverClock]
		// resources of type target; begin of array
		// ID always present to identify changed resource
		// changed field or property
		// no other field or property changed since last update
		
		// ID shows that this resource still exists but is unchanged
		
		// null means that entire list was not changed since last update
		// also events were not changed
		// same for variables but
		// resources of type script were changed
		// resource with ID only to show existence
		
		// ID of a changed resource
		// changed field or property
		
		// ID of a changed resource
		// changed field or property
		
		// no CRUD operations for segments since last update
		// empty array: entire list was just deleted!
		// models were not changed
		// also users stay the same
		// and finally setting too