Rocket® Data Replicate and Sync REST API V7.0

Notices

Copyright

Trademarks

Examples

License agreement

Corporate information

Rocket Software, Inc. develops enterprise infrastructure products in four key areas: storage, networks, and compliance; database servers and tools; business information and analytics; and application development, integration, and modernization.

Website: www.rocketsoftware.com

Rocket Global Headquarters

77 4th Avenue, Suite 100

Waltham, MA 02451-1468

USA

To contact Rocket Software by telephone for any reason, including obtaining pre-sales information and technical support, use one of the following telephone numbers.

Country and Toll-free telephone number

- United States: 1-855-577-4323
- Australia: 1-800-823-405
- Belgium: 0800-266-65
- Canada: 1-855-577-4323
- China: 400-120-9242
- France: 08-05-08-05-62
- Germany: 0800-180-0882
- Italy: 800-878-295
- Japan: 0800-170-5464
- Netherlands: 0-800-022-2961
- New Zealand: 0800-003210
- South Africa: 0-800-980-818
- United Kingdom: 0800-520-0439

Contacting Technical Support

The Rocket Community is the primary method of obtaining support. If you have current support and maintenance agreements with Rocket Software, you can access the Rocket Community and report a problem, download an update, or read answers to FAQs. To log in to the Rocket Community or to request a Rocket Community account, go to [www.rocketsoftware.com/support.](www.rocketsoftware.com/support) In addition to using the Rocket Community to obtain support, you can use one of the telephone numbers that are listed above or send an email to [support@rocketsoftware.com.](mailto:support@rocketsoftware.com)

Table of contents

Rocket® Data Replicate and Sync REST API

For command line usage, i.e.: for automation purposes via scripts or monitoring use cases, Rocket® Data Replicate and Sync has a *REST API*. The application is implemented as a JAVA class.

REST is an abbreviation for *Representational State Transfer* and is an abstraction of structure and behavior of the *World Wide Web* (WWW). Its purpose is changing the states of an application via data transfer.

The REST API implementation acts as a client (just like the Dashboard) to a Rocket® Data Replicate and Sync agent. Hence, a connection to an agent is necessary in order to work. The REST API implementation uses the Dashboard's classes. Thus, the `JAR` file `Dashboard.jar` is needed, however, the Dashboard itself is not started. A parallel use of the REST API and the Dashboard (as GUI) is possible.

The REST API's in- and output is *JSON* (JavaScript Object Notation),an easily human readable text format (https://json.org/). Access to the REST API is realized via the stateless application level protocol *HTTP* (HyperText Transfer Protocol).Therefore, REST API commands can not be sent via command line, but need a HTTP capable application.

The multi-platform tool, `curl` (Client for URLs), is very well suited for testing purposes. With `libcurl`, a library with C API is provided. A example for a REST API call via `curl`:

```
curl -H "@auth"-X POST http://127.0.0.1:8080/DataSources -d @MySQL.txt
```

POST is the HTTP method called in the so-called context `/DataSources`. Here, the REST API is running on localhost and the TCP/IP listener is set to port 8080. The text file `MySQL.txt` is transferred to `cURL` by the -d option, user name and password in the plain text file auth by -H.

If several queries are specified, they must be separated with `&` and the entire URL must be set in quotation marks with curl:

```
curl-H "@auth" -X GET "http://127.0.0.1:8080/DataSources?Agent=Agent1&All=y"
```

Note: For shells: Contrary to `Bash` (Bourne-again shell) and some others, in certain command-line interpreters, several special characters like `?` and `=` need to be escaped by putting a backslash (`\`) in front.

Starting and stopping the REST API server

To start the REST API, enter:

```
java -cp ./Dashboard.jar org.bos.RESTApiServer [OPTIONS]
java -cp ./Dashboard.jar org.bos.RESTApiServer -connect=192.168.0.26:4135
```

A launch with the option `-help` leads to a short description of the possible parameters. All supported options are described in the following table. Each parameter can only be stated once.

Option	Description/Example
-cfg	This parameter states a configuration file. All following options can be committed by this file instead of passing them as separate parameters. Please state the fully-qualified domain name (*FQDN*). In case of the file containing blanks, it must be encompassed in quotes. Example:

Option	Description/Example
	`-cfg=/home/bob/RDRS/7/bin/REST-API.ini`
-port	Specification of the TCP/IP port on which the REST API should be accessible (not the port of the agent, this is specified with -connect). If this option is not set, the port is set to 8080.
-connect	To indicate a string for establishing a connection to a Rocket® Data Replicate and Sync agent. The syntax is *`username/password@host:port`* . The port is set in the file `tcAgent.ini` (`Port=<port>`). It is important to ensure that the SSL port is also used for an SSL connection (`SSLPort=<ssl port>`). Example: `-connect=bob/password@sunsparc.bos.net:4120`
-keystore	To set an optional SSL keystore file and its password. If set, an SSL connection to agent stated by -connect will be established. The syntax is *`keystorefile;password`* . Example: `-keystore=/home/bob/SSL/truststore.pfx;pwd`
-trace	With -trace=y, the creation of a trace file is activated. It is written to the log directory of Rocket® Data Replicate and Sync 7, which is set by `ProtocolDir=` in the file `tcAgent.ini`. The filename is structured in the following way: `REST_API-`*`<AgentName>`*`-`*`<DateTime(zone time)>`*`.txt`
-version	To print the module version
-encoding	To indicate the character encoding (e. g. UTF-8)
-linebreaks	By `-linebreaks=y`, line breaks in the JSON output will not be removed. The JSON library which is used introduces line breaks in some places. The JSON syntax does not require any and they can lead to problems (for instance, with Kafka). Hence, the default is removing them.
-https_keystore	By this parameter, a SSL Keystore file and password can be stated. If that file is stated, REST API calls cann be executed via HTTPS. The syntax is the same as for `-keystore` above
-https_port	Specification of the TCP/IP port on which the REST API should be accessible via HTTPS (not the port of the Agent, this is specified with `-connect`). This only works with denotion of `-https_keystore`.
-https_only	If this parameter is set to n (`-https_only=n`), when using `-https_keystore`, aside from HTTPS calls, insecure HTTP calls are also allowed.
-https_protomin	To set the lowest HTTPS protocol version
-https_protomax	To set the highest HTTPS protocol version
-https_provider	To indicate the provider for HTTPS

It is possible to state a configuration file by -cfg. If the file does not exist, it will be created after a successful REST API start. Therefore, at least the parameter -connect must be used. From the following start on, the newly created configuration file will be used. The parameter `-cfg=`*`<Path/to/configuration file>`* will suffice then.

The start of the REST API is accompanied by the output of more information:

- the host on which it is running

- the port it is listening to (HTTP and HTTPS)
- the IP address and port of the agent it is directly connected to
- if this connection is rendered via SSL or not
- if a configuration file is used and if yes, its path and file name
- if a trace is written and if yes, its path and file name
- date and time of REST API startup.
- an example call and
- the available contexts

Notes to REST API output

Aside from the HTTP status code, REST API prints its output in JSON (JavaScript Object Notation) format.

The JSON output does not contain any line breaks or empty lines.

Using a JSON parser, it is possible to reshape the JSON output more human readable.

Example of a direct JSON output (shortened):

```
{"REST API is running":true,"Host IP":"127.0.0.1","Write trace file":true,"
Trace file":"\/home\/user\/RDRS\/log\/REST_API-Agent-Linux-<…>.txt"}
```

A parser reshapes the output by including newline, whitespace and possibly tabs, too.

```
{
   "Host IP" : "127.0.0.1",
   "REST API is running" : true,
   "Trace file" : "/home/user/RDRS/log/REST_API-Agent-Linux-<…>.txt",
   "Write trace file" : true
}
```

The slash (/) as a directory separator under Linux or for example as Table_Type DB2/MVS gets escaped as \/ in the direct output. A JSON parser depicts it correctly as/, see above.

Otherwise, special characters are escaped within the JSON output and also need to be escaped when reading JSON input data:

Backspace	\b
Form feed	\f
Newline	\n
Carriage return	\r
Tab	\t
Double quote	\"
Backslash	\\

Built-in help (OPTIONS)

For each resource, there is aa HTTP method `OPTIONS`. It lists all of the resources' available methods, as well as a short description. For instance, the call of `OPTIONS` on `Agents`:

```
curl -H "@auth" -iX OPTIONS http://127.0.0.1:8080/Agents
HTTP/1.1 200 OK

[
{"= HTTP method =              ": "= Explanation =",
"GET /Agents/defined          ": "Print information on all defined RDRS agents",
"GET /Agents/defined/<Name>   ": "Print information on a defined RDRS agent",
"GET /Agents/connected        ": "Print information on connection status of all RDRS agents",
"GET /Agents/connected/<Name> ": "Print information on connection status of an RDRS agent",
"= Optional queries =  ": "= Explanation =",
"?Group=<GroupID>      ": "Use <GroupID> (default value = 'DEFAULT')"}
]
```

General information about HTTP method calls

Each method call is executed by the specification of an HTTP method (e. g. `GET`) and a resource (e. g. `/DataSources`).

If authentication is enabled in the Rocket® Data Replicate and Sync Agent, the header must encompass a user name and a password. Additional data must be transferred for some methods, such as creating a data source.

For a refined call, it is possible to state special statements via queries, such as the group of repository information.

The following chapters list the resources with their HTTP methods. Correct execution is described first, followed by possible problem cases. The calls are displayed by the `curl` command on the command line, the output contains the HTTP status code and the data in JSON format.

Sub resources

Some resources require a supplementary sub resource after the slash (`/`). For instance, the sub resource `/defined` is needed after `/Processes` in order to handle defined processes. Otherwise, status code `400 Bad Request` is returned along with a list of possible sub resources (see "400 Bad Request - incorrect method call").

Queries

Query 'Groupid' (Repository Groups)

Rocket® Data Replicate and Sync is client-capable, which means that logical subdivisions can be defined in the repository without them having mutual insight. Here, such a part is called repository group, or simply group, for short. A group is identified by its GroupID.

In a method call, a group is declared by the query `Groupid=<GroupID>` .

Example:

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/terminated?Groupid="TestEnvironment"
```

If a query for a groupid is not set, the group `DEFAULT` is taken, which exists in every repository. A groupid query may be set in every resource, yet it is ignored if its indication makes no sense (e. g. in resource `/RestApi`).

Queries ValidFrom and ValidUntil (timely validity)

The queries `ValidFrom` and `ValidUntil` allow the output of only those resources that are valid in the period defined by them. This means that all resources are found whose `ValidFrom` is less than or equal to the `ValidFrom` specified here, and whose `ValidUntil` is greater than or equal to the `ValidUntil` specified here.

These queries do not apply for the resources `/Agents`, `/Processes/terminated`, `/Processes/active` and `/Processes/all`.

Without these queries, the current time is always selected internally for `validFrom` in `GET`, as well as 9999-99-99-99.99.99.999999 for `validUntil`. This means that only the entries that are valid at the current time of the method call are output.

Example:

```
curl -H "@auth" -iX GET "http://192.168.0.26:8080/Processes/
defined?ValidFrom=2023-05-11-22.59.59.000000&ValidUntil=2030-05-11-22.59.59.000000"
```

`ValidFrom` must be ≤ `ValidUntil`, otherwise the following error message is the output:

```
HTTP/1.1 400 Bad Request

{"ERROR": "Invalid time frame: Query ValidFrom > Query ValidUntil"}
```

Query 'All' (ignore timely validity)

By stating the query `All=y` , all entries are output, regardless of their timely validity, when using `GET`. The values of the queries `ValidFrom` and `ValidUntil` are ignored, if stated.

Query 'Overview" (overview)

The query `Overview=y` allows the output of only the most important key-value pairs in the JSON output. For example, for `/DataSources` it is only `Name`, `ValidFrom` and `ValidUntil`. This allows a clear output.

Query 'SimpleOutput' (simplified output)

For the same reason, the query `SimpleOutput=y` exists. The parameter `validFrom` and `validUntil` are output only if they differ from 0000-00-00-00.00.00.000000 or 9999-99-99-99.99.99.999999, respectively.

Call error: "Failed to connect" or similar

If calling the REST API is not possible, the cause may be that the REST API is not running or not accessible in the network.

The error output depends on the program calling the REST API: For example, curl outputs this error message:

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/Agents/defined
curl: (7) Failed to connect to 127.0.0.1 port 8080 after 0 ms: Connection refused
```

Possible general HTTP return codes

Some return codes are independent of the resource and HTTP method. Therefore, they are no longer listed again in the following chapters.

200 OK - regular execution

The HTTP method was correctly called with valid parameters and carried out properly.

Note: For `POST /DataSources`, there is yet another HTTP status code indicating a proper execution: `201 Created`.

400 Bad Request - incorrect method call

If an argument is essential, but none given:

```
 curl -d @MySQL_DS -H "@auth" -iX PUT http://127.0.0.1:8080/DataSources/
HTTP/1.1 400 Bad Request

[
{"ERROR": "No argument with name of data source provided"}
]
```

If a mandatory subresource was omitted:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Agents
HTTP/1.1 400 Bad Request

{"ERROR": "No subresource stated after /Agents/, must be 'defined' or 'connected'"}
```

If an invalid subresource was given:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Agents/defin
HTTP/1.1 400 Bad Request

{"ERROR": "Subresource 'defin' is not 'defined' or 'connected'"}
```

If the stated HTTP method is nonexistent:

```
 curl -H "@auth" -iX GETT http://127.0.0.1:8080/RestApi
HTTP/1.1 400 Bad Request

{"ERROR": "Method 'GETT' is not supported"}
```

If the given resource is nonexistent:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Ag
HTTP/1.1 400 Bad Request

{"ERROR": "Malformed URI"}
```

If a necessary parameter is missing at HTTP method call (here: stating username/password necessary for LDAP):

```

{"ERROR": "Error connecting to agent (no message).",
"Last returncode": "0000000C",
"Returncode explanation": "A required parameter is missing.",
"INFORMATION": "The required missing parameter might be username/password."}
```

401 Unauthorized - no access to the agent

If the REST API was started using an LDAP connection, username and password need to be stated at every REST API call. If not correctly given, this output results:

```
HTTP/1.1 401 Unauthorized
{"ERROR": "Error connecting to agent (no message).",
"Last returncode": "08100005",
"Returncode explanation": "An LDAP error occurred. The username/password is incorrect."}
```

503 Service Unavailable - agent not available

If the agent is not running anymore (or no network access to it anymore), this output results:

```
HTTP/1.1 503 Service Unavailable
[
{"ERROR": "Error connecting to agent. Received message: "java.net.ConnectException: Connection r
efused",
"INFORMATION": "Connection error for agent on address '192.168.0.26', port 4135. Please start th
e agent on the machine with that address or change the address in the preferences"
}
]
```

It specifies which IP address the machine on which the agent is running has and which port it should listen on. It is always the agent, to which the REST API directly connects to (this is also stated in the -connect-String).

Overview: Resources and their HTTP methods

`OPTIONS`	`/<resource>`	Help/method overview to the resource
`GET`	`/RestApi`	Print status information
`STOP`	`/RestApi`	Terminate REST API
`GET`	`/Agents/defined`	List defined agents for the agent network
`GET`	`/Agents/defined/<Name>`	Print stated defined agent
`GET`	`/Agents/active`	List active agents
`GET`	`/Agents/active/<Name>`	Print stated active agent
`DELETE`	`/Agents/defined/<Name>`	Delete stated defined agent
`GET`	`/Groups`	List of all groups
`GET`	`/DataSources`	List all data sources
`GET`	`/DataSources/<Name>`	Print stated data source
`RENAME`	`/DataSources/<Name>/<NewName>`	Rename a stated data source
`SAVEAS`	`/DataSources/<Name>/<NewName>`	Save a copy of a stated data source
`POST`	`/DataSources`	Add or replace a data source
`PUT`	`/DataSources/<Name>`	Change stated data source
`DELETE`	`/DataSources/<Name>`	Delete stated data source
`GET`	`/OutputTargets`	Print all output targets
`GET`	`/OutputTargets/<Name>`	Print stated output target
`POST`	`/OutputTargets`	Add or replace an output target
`PUT`	`/OutputTargets/<Name>`	Change stated output target
`DELETE`	`/OutputTargets/<Name>`	Delete stated output target
`GET`	`/Processes/defined`	List all defined processes
`GET`	`/Processes/defined/<Name>`	Printstated defined process
`RENAME`	`/Processes/defined/<Name>/<NewName>`	Rename stated defined process
`START`	`/Processes/defined/<Name>`	Start statedprocess
`GET`	`/Processes/active`	List all running processes
`GET`	`/Processes/active/<Name>`	Print stated running process
`GET`	`/Processes/`	Print running process matching stated Name and EndTime

	`active/<Name>/<EndTime>`	
`GET`	`/Processes/terminated/`	List all terminated processes
`GET`	`/Processes/terminated/<Name>`	Print stated terminated process
`GET`	`/Processes/terminated/<Name>/<EndTime>`	Print finished process matching stated Name and EndTime
`GET`	`/Processes/all`	List all processes (active and terminated)
`GET`	`/Processes/all/<Name>`	Print stated process (active or terminated)
`GET`	`/Processes/all/<Name>/<EndTime>`	Print process (running or terminated) matching stated Name and EndTime
`GET`	`/Processes/rexxvar/<Name>/<PID>/<REXXVar>`	Print stated REXX variable of a process stated by Name and ProcessID
`GET`	`/Processes/latency/<Name>/<PID>`	Print latency of a running process stated by Name and ProcessID
`STOP`	`/Process/<Name>/<PID>`	Terminate a process stated by Name and ProcessID
`DELETE`	`/Processes/defined/<Name>`	Delete a process definition matching given Name
`DELETE`	`/Processes/terminated/<Name>/<EndTime>`	Delete a terminated process (history entry) matching given Name and EndTime
`GET`	`/Jobs`	Print all jobs
`GET`	`/Jobs/<Name>`	Print stated job
`RENAME`	`/Jobs/<Name>/<NewName>`	Rename stated job
`SAVEAS`	`/Jobs/<Name>/<NewName>`	Save a copy of the stated job
`START`	`/Jobs/<Name>`	Activate stated job
`DELETE`	`/Jobs/<Name>`	Delete stated job
`GET`	`InputObjects`	Print input tables (including associated dependencies)
`GET`	`OutputObjects`	Print output tables (including associated dependencies)
`GET`	`InputFields`	Print input fields
`GET`	`OutputFields`	Print output fields
`GET`	`ReplicationObjects`	Print table and field connections
`GET`	`RequestManagements`	Print RequestManagement entries
`START`	`Import`	Start an import
`SENDFILE`	`Import`	Send file for import
`DELETEFILE`	`Import`	Delete file for import
`CREATE`	`Export`	Create export commands

`RUN`	`Export`	Execute export commands
`INIT`	`Tool`	Initialize the repository
`BACKUP`	`Tool`	Create repository backup
`RESTORE`	`Tool`	Restore the repository backup
`START`	`Tool`	Start a Process
`START`	`Migrate`	Migrate the repository
`GET`	`/`	Return a link forwarding to online documentation

Resource RestAPI

OPTION /RestApi → Help / methods overview

```
 curl -H "@auth" -iX OPTIONS http://127.0.0.1:8080/RestApi
HTTP/1.1 200 OK

[
{"= HTTP method =": "= Explanation =",
"GET  /RestApi  ": "Print REST API status information",
"STOP /RestApi  ": "Exit REST API"}
]
```

GET /RestApi → Status information

The `GET` method called upon the resource `/RestApi` is reporting the REST API's status:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/RestApi
HTTP/1.1 200 OK

[
{"Status": "REST API is running",
"REST API Start Time (Zone Time): ": "2023-05-15-09.38.49.384000",
"REST API Start Time (UTC): ": "2023-05-15-07.38.49.384000",
"REST API Host IP": "127.0.0.1",
"REST API Host FQDN": "localhost",
"REST API Port": 8080,
"REST API Revision ": "1229",
"Repository Agent Connection": "192.168.0.26:4135",
"Agent Version": "tcSCRIPT Unicode 7.0.1603, (MTR), 64-bit, tcREXX 1.1.487"}
]
```

STOP /RestApi → Exiting REST API

```
 curl -H "@auth" -iX STOP http://127.0.0.1:8080/RestApi
HTTP/1.1 200 OK

[
{"Status": "REST API has exited"}
]
```

401 Unauthorized - missing authorization to exit REST API

If `SYSTEM.SHUTDOWN` is not authorized for the connected user, REST API will not exit. Details on this are depicted in "Authentication".

The following output results:

```
 curl -H "@auth" -iX STOP http://127.0.0.1:8080/RestApi
HTTP/1.1 401 Unauthorized

{"ERROR": "An LDAP error occurred. The given user could not be found or has not been permitted f
or this action.",
"Status": "REST API is still running"}
```

Resource Agents

OPTIONS /Agents → Help / method overview

```
 curl -H "@auth" -iX OPTIONS http://127.0.0.1:8080/Agents
HTTP/1.1 200 OK
[
{"= HTTP method =              ": "= Explanation =",
"GET /Agents/defined          ": "Print information on all defined
                                    Rocket® Data Replicate and Sync agents",
"GET /Agents/defined/<Name>   ": "Print information on a defined Rocket® Data Replicate and Syn
c agent",
"GET /Agents/active    ": "Print information on status
                          of all Rocket® Data Replicate and Sync agents",
"GET /Agents/active/<Name> ": "Print information on status
                             of a Rocket® Data Replicate and Sync agent",
"DELETE /Agents/defined/<Name>": "Delete a defined Rocket® Data Replicate and Sync agent",
"= Optional queries =         ": "= Explanation =",
"?Group=<GroupID>             ": "Use <GroupID> (default value = 'DEFAULT')"}
]
```

GET /Agents/defined → List defined agents

List all defined agents:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Agents/defined
HTTP/1.1 200 OK

{"Results": [
{"AgentName": "Agent_Linux",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
"IPAddress:Port": "192.168.0.26:4135",
"Platform": "Linux",
"SSLConnect": "No",
"AutoConnect": "Yes",
"UserID": "",
"Password": "",
"CertDef": ""},
{"AgentName": "Agent_Win",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
"IPAddress:Port": "192.168.0.46:4135",
"Platform": "MSWIN",
"SSLConnect": "No",
"AutoConnect": "Yes",
"UserID": "",
"Password": "",
"CertDef": ""}]}
```

If an agent’s name is given as an argument, only this one will be printed:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Agents/defined/Agent_Linux
HTTP/1.1 200 OK

{"Results": [
{"Agent": "Agent_Linux",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
"IPAddress:Port": "192.168.0.26:4135",
"Platform": "Linux",
"SSLConnect": "No",
"AutoConnect": "Yes",
"UserID": "",
"Password": "",
"CertDef": ""}]}
```

If no agents are defined:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Agents/defined
HTTP/1.1 200 OK

[
{"Information", "No agents defined."}
]
```

404 Not Found - stated agent not found

If the stated agent does not exist:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Agents/defined/Agent_L
HTTP/1.1 404 Not Found

[
{"ERROR": "Agent 'Agent_L' not found."}
]
```

GET /Agents/active → Print the agents' status

Lists all agents.

Important: The execution of this method takes some seconds, and within larger agent networks presumably longer, because connections to all defined agents will be established, at least the attempt will be made.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/Agents/active
HTTP/1.1 200 OK
{
"Results":  [
  {"AgentName":  "TCVA-AMSDEVRDRSFBW01",
   "Connected":  true,
   "Version":  "7.0.0",
   "Platform":  "MSWIN",
   "LocalTime":  "2025-02-24-09.02.58.694412",
   "UTCTime":  "2025-02-24-08.02.58.694419",
   "RelDat":  "2025-02-20",
   "Security":  "N",
   "CCSID":  1252,
   "Revision":  2601,
   "DefaultLanguage":  "EN",
   "Job":  "TCVM",
   "JobId":  "00001FB0",
   "StructureRepository":  "POSTGRESQLhost=127.0.0.1;port=5432;dbname=postgres;user=postgres;pas
sword=xxx;",
   "MailServer":  "local.mailserver.com:25",
   "SecParm":  "",
   "TCPIP":  "",
   "Workstation":  "workstation.name",
   "Trace":  true,
   "TraceFile":  "C:\\RDRS\\log\\TCVA-AMSDEVRDRSFBW01-2025-02-21-09.54.52.745324-trac",
   "ClassDefFile":  "C:\\RDRS\\Config\\ClassDef.File",
   "AutostartFile":  "C:\\RDRS\\autostart.file",
   "ScheduleFile":  "C:\\RDRS\\scheduler.file",
   "ConfDirectory":  "C:\\RDRS\\Config\\",
   "ScriptDirectory":  "C:\\RDRS\\Scripts\\",
   "SupportedDatabases":  "BigData, DB2 (DRDA), DB2 (Client), Informix, MSSQL, MySQL, MariaDB, O
DBC, Oracle, PostgreSQL, JavaVM for BigQuery"},
  {"AgentName":  "RS28-Agent",
   "Connected":  true,
   "Version":  "7.0.0",
   "Platform":  "z/OS",
   "LocalTime":  "2025-02-24-03.03.01.061151",
   "UTCTime":  "2025-02-24-08.03.01.061153",
   "RelDat":  "2025-02-20",
   "Security":  "N",
   "CCSID":  37,
   "Revision":  2601,
   "DefaultLanguage":  "EN",
   "Job":  "RDRS3463",
   "JobId":  "E0F73D00",
   "ManagerLoadLib":  "DEV.RDRS700.LOADLIB",
   "StructureRepository":  "MANAGER   TCVA-AMSDEVRDRSFBW01;IP=amsdevrdrsfbw01:4135",
   "MailServer":  "",
   "SecParm":  "",
   "TCPIP":  "",
   "CPU-ID":  "XXX",
   "Trace":  true,
   "TraceFile":  "T0H36HVG (JES)",
   "ClassDefFile":  "TVSMDSK:CONFIG:/ClassDef.File",
   "AutostartFile":  "TVSMDSK:CONFIG:/Autostart.File",
   "ScheduleFile":  "TVSMDSK:CONFIG:/Scheduler.File",
   "ConfDirectory":  "TVSMDSK:CONFIG:/",
   "ScriptDirectory":  "TVSMDSK:CONFIG:/Scripts/",
   "SupportedDatabases":  "Adabas, DB2 (DRDA), DB2 (CAF), IDMS, IMS/DB, VSAM",
   "DB2SubSystems":  "DB9D,UC1A",
   "IMSIDs":  "IFA4",
   "AdabasSVCs":  "249,250",
   "IDMSCVs":  "CV185"},
  {"AgentName":  "TCVA-AMSDEVRDRSFBL01",
   "Connected":  false}]
}
```

Similarly, an agent can also be specified explicitly, then only this is output (with the same HTTP status code):

`curl -H "@auth" -iX GET http://127.0.0.1:8080/Agents/active/RS28-Agent`

As the output shows, the connection status to each agent will be printed ("`Connected`": "`true`" or "`false`"). If there is a connection, detailed information of this agent will be listed.

Important: If an agent is not connected at the time of the method call, but is running and accessible, it will be connected due to this call. Nonetheless, it will be reported as `not connected` ("`Connected`": "`false`"), as it was the case at the time of the call. Called anew, the method now returns the result "`Connected`": "`Y`" as well as detailed information on the agent (since the connection existed at the time of the second call).

404 Not Found - the stated Agent is nonexistent

If the stated agent is nonexistent:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Agents/active/Agent_L
HTTP/1.1 404 Not Found
{"ERROR": "Agent 'Agent_L' not found."}
```

POST Agents/defined → Add an Agent

The method POST Agents/defined enables adding an existing agent to the agent network.

`curl -d @post_agents.json -iX POST localhost:8080/Agents/defined`

```
HTTP/1.1 201 Created
{"Success":"Agent Agent-Linux added (10.215.10.44:4135)"}
```

A minimal JSON input file is:

```
{
  "Agents": [
    {
      "IPAddress": "10.215.8.51:4131"
    }
  ]
}
```

You can use the following keys:

`ValidFrom`, `ValidUntil`, `Description`, `Userid`, `Password`, `CertDef` and `SSLConnect`.

The values for all the keys are string. The value for `SSLConnect` must be `N` or `Y`, `CertDef` states `a .CertDef` file in the Config sub-directory of Rocket® Data Replicate and Sync.

Note: There is no key `AgentName`, since the agent's name is returned by the agent itself during this method call.

201 Created - New agent was successfully added to the agent network

```
HTTP/1.1 201 Created
{"Success":"Agent Agent-Windows added (10.215.8.51:4131)"}
```

400 Bad Request - Stated agent is already known

```
HTTP/1.1 400 Bad Request
{"Error":"The agent Agent-Windows is already part of the agent network."}
```

DELETE /Agents/defined → Delete a defined agent

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/Agents/defined/Agent_new
HTTP/1.1 200 OK

[
{"Success": "Agent 'Agent_new' deleted"}
]
```

412 Precondition Failed - Agent is still referenced

If the agent is still referenced in process definitions, data sources, input tables, output tables, output targets or jobs, it can not be deleted. Therefore, it must be removed in the references first. The following output is generated, with indication of the references still present:

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/Agents/defined/Agent_new
HTTP/1.1 412 Precondition Failed

{"Error": "Agent 'Agent_new' was NOT deleted",
"Information": "The Agent 'Agent_new' is still referenced in the following resource(s) (Type of
resource: [Name, ...])",
"Process Definitions": [
"BULK_DB2_2_MARIA",
"BULK_ORA_2_PG",
],
"Data Sources": [
"Source_DB2"
"Source_Oracle",
],
"Input Tables": [
"DB2/MVS.DBCG.DEMO.ARTICLE",
"ORACLE.ORCL.system.ARTICLE"
],
"Output Targets": [
"Target_MARIA",
"Target_PG",
],
"Output Tables": [
"Target_MARIA.test.ARTICLE",
"Target_PG.public.ARTICLE",
],
"Jobs": [
"DO_BULK"
],
"System Checks": [
"Internal check id 6332220"]}
```

Resource Groups

Using the resource Groups repository definitions of table Groups can be retrieved, created, changed and deleted.

OPTIONS Groups → Method Overview

```
curl -H "@auth" -iX OPTIONS http://127.0.0.1:8080/Groups

HTTP/1.1 200 OK

{"= HTTP method =              ":"= Explanation =",
"GET    /Groups               ":"Get information for Group(s)",
"POST   /Groups               ":"Add a new Group",
"PUT    /Groups               ":"Change a Group",
"DELETE /Groups               ":"Delete a Group",
"= Optional queries for GET = ":"= Explanation =",
"groupid=<UniqueId>           ":"Return Group with group id <GroupId> only",
"details=<Y|N>                ":"Returned result: 'N' key fields only, 'Y' all fields (default v
alue = 'N')",
"= Request data =             ":"= Explanation =",
"Data in JSON format          ":"Use data to specify the attributes of the Group(s) to post/put/
delete"}
```

GET Groups → Information about Groups

The simple call without further query parameters returns all entries of table ‘Groups’.

The result is returned in JSON notation. It informs whether the output is with all details or not and the individual Group entries as JSON objects within a JSON array called ‘Results’. The key names of the key-value pairs in these objects correspond to the names of the fields of the repository objects (here, for example, the field name 'GroupId' of the repository table 'Groups').

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/Groups

HTTP/1.1 200 OK

{"Details":false,"Results":[
{"GroupId":"DEFAULT"},
{"GroupId":"Migrate_Dest"}
]}
```

The call can be restricted by specifying an additional parameter in the query data. Only the Group entries that match the criteria of this parameter are returned.

The following parameters are available:

Parameter	Description
groupid	Restricts the determined entries to a specific group

Parameter	Description
details	Used to set the level of detail of the return. The possible values are: • Y: all fields of the repository object • N: only the key fields of the repository object (default)

Notice:

If the restrictive parameter ('groupid') is set, 'Y' is automatically used for 'details'. In this case the result only consists of one entry with all the information from the repository definition of this entry.

Example of using the parameter

Only the repository entries with groupid ‘Migrate_Dest’ should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/Groups?groupid=Migrate_Dest

HTTP/1.1 200 OK

{"Details":true,"Results":[
{"GroupId":"Migrate_Dest",
"Description":"Group for testing migration"}
]}
```

POST Groups → Inserting Group entries

The call allows Group entries to be inserted into the repository.

For the insert all values of a Group entry must be stored in a file in JSON format, and this file is then sent in the request data. The names of the values correspond to the field names in repository table “Groups”, and are not case sensitive.

The individual Group entries are embedded within the JSON array “Groups”, this identifier is also not case sensitive.

Example

An entry is to be inserted into the repository table Group. To do this, a JSON file “data.json” with the following content is created.

```
{
"Groups": [
{"GroupId":"MyNewGroup",
"Description":"Description of MyNewGroup"
}
]
}
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/Groups"
```

```
HTTP/1.1 200 OK

{"Groups posted":[
{"GroupId":"MyNewGroup"}
]}
```

PUT Groups → Updating Group entries

Calling this method allows to update Group entries in the repository.

To update, the key values of the entry to be updated as well as all values of the resulting entry must be saved in a file in JSON format, and this file is then sent in the request data. The names of the values correspond to the field names in repository table "Groups", and are not case sensitive. The names of the key values of the entry to be updated is preceded by the prefix "Before_".

The individual Group entries are embedded within the JSON array "Groups", this identifier is also not case sensitive.

Example

An entry is to be updated in the repository table Groups. To do this, a JSON file "data.json" with the following content is created.

```
{
"Groups": [
{"Before_GroupId":"MyNewGroup",
"GroupId":"MyNewGroup",
"Description":"Updated description"
}
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/Groups"

HTTP/1.1 200 OK

{"Groups putted":[
{"GroupId":"MyNewGroup"}
]}
```

DELETE Groups → Deleting Group entries

Calling this method allows to delete Group entries from the repository.

To delete, the key values of the entry to be deleted must be saved in a file in JSON format, and this file is then sent in the request data. The names of the values correspond to the field names in repository table "Groups", and are not case sensitive.

The individual Group entries are embedded within the JSON array "Groups", this identifier is also not case sensitive.

Example
An entry is to be deleted from the repository table Groups. To do this, a JSON file "data.json" with the following

content is created.

```
{
"Groups": [
{"GroupId":"MyNewGroup"
}
] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX DELETE "http://127.0.0.1:8080/Groups"

HTTP/1.1 200 OK

{"Groups deleted":[
{"GroupId":"MyNewGroup"}
]}
```

Error messages for resource Groups

If an error occurs when executing an option of resource Groups, an HTTP return code not equal to 200 is returned.

The following is a brief overview of possible error messages and their causes:

```
curl -H "@auth" -iX GETT http://127.0.0.1:8080/Groups

HTTP/1.1 400 Bad Request

{"Error": "Method 'GETT' is not supported"}
```

An incorrect method was used in the call. The call needs to be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/Groups ...
HTTP/1.1 500 Internal Server Error

{"Error": " Groups failed",
"Reason": "Caught exception while creating result",
"Exception": "<Exception message>"}
```

If there is a problem with the internal processing of a call, this message will be returned. The placeholder <Exception message> is replaced by a message that corresponds to the problem.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/Groups"
HTTP/1.1 500 Internal Server Error

{"Error":"Groups failed",
"Reason":"Got SQL error message while inserting ",
"Object":"Group MyNewGroup (2024-07-04-12.34.56.000000)",
"SQL message":"TCS0597E;70R1940,WIN64_6.2.9200; Repository SQL error: TCS0256E;70R1940,WIN6
4_6.2.9200; PostgreSQL libpq SQLSTATE 23505, ERROR: duplicate key value violates unique constrai
nt »groups_pkey1« DETAIL: key »(groupid, validfrom, validuntil)=(MyNewGroup, 0000-00-00-00.00.0
0.000000, 9999-99-99-99.99.99.999999)« already exists.; N\/A."}
```

If there are problems (SQL errors) when inserting, updating or deleting objects in the repository, this message is returned. The affected object is given as well as the error message received from the database.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/Groups"
HTTP/1.1 400 Bad Request

{"Error":"Groups failed",
"Reason":"Updating 'Groups' object in repository not possible, because not all key values of 'Be
fore' object have been specified.",
"Object":"Group MyNewGroup"}
```

If not all key values are provided to uniquely identify an object, this will be indicated with a message like this.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/Groups"
HTTP/1.1 400 Bad Request

{"Error":"Groups failed",
"Reason":"Updating 'Groups' object in repository not possible, because object does not exist in
repository.",
"Object":"Group MyNewGroup2"}
```

If no object matching the key values passed is found in the repository, this will be indicated with a message like this.

Resource DataSources

The resource DataSources enables to print, create and modify data sources.

OPTIONS /DataSources → Help / method overview

```
curl -H "@auth" -iX OPTIONS http://127.0.0.1:8080/DataSources
HTTP/1.1 200 OK

{"= HTTP method =               ": "= Explanation =",
"GET    DataSources             ": "Print information on all Data Sources",
"GET    DataSources/<Name>      ": "Print information on a Data Source",
"RENAME DataSources/<Old>/<New> ": "Rename a Data Source",
"SAVEAS DataSources/<Old>/<New> ": "Save a copy of a Data Source",
"POST   DataSources             ": "Add a new or replace a Data Source",
"PUT    DataSources             ": "Change a Data Source",
"DELETE DataSources/<Name>      ": "Delete a Data Source",
"= Optional queries =           ": "= Explanation =",
"groupid=<GroupId>              ": "Use <GroupId> (default value = 'DEFAULT')",
"validfrom=<PointInTime>        ": "Consider Data Sources valid from <PointInTime>",
"validuntil=<PointInTime>       ": "Consider Data Sources valid until <PointInTime>"}
```

GET /DataSources → Print information on data sources

List all data sources:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/DataSources
HTTP/1.1 200 OK
Date: Mon, 18 Jul 2022 14:15:57 GMT
Content-length: 1288

{"Results": [
{"SourceName": "DB2",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"SourceType": "DB2",
"Description": "",
"AgentName": "Agent_Linux",
"ConnectionString": "HOST=192.168.0.223;PORT=5040;DATABASE=DALLASC;PLAN=TCEXPRESS;PACKAGE=TCELF0
00;UID=FRANK;PWD=xxxxxxxxxxxxxxxxxxx;SERVERCLASS=M;SUBSYS=DBCG"},
{"SourceName": "PG",
"SourceType": "POSTGRESQL",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
"AgentName": "Agent_Linux",
"ConnectionString": "host=192.168.0.26;port=5432;dbname=postgres;user=postgres;password=xxxxxxxx
xxxxxxxxxxx"}]}
```

By stating a specific data source, only this one will be reported:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/DataSources/DB2
```

If no data sources are defined:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/DataSources/
HTTP/1.1 200 OK

{"INFORMATION": "No defined data sources"}
```

404 Not Found - stated data source not found

If the given data source does not exist:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/DataSources/DB
HTTP/1.1 404 Not Found

[
{"ERROR": "Data source 'DB' not found. Check correct spelling."}
]
```

Note: This output also results if no data sources are defined, but a data source was stated in the call.

RENAME /DataSources → Rename a data source

Renaming a data source requires stating the old and new data source name:

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/DataSources/DS_old/DS_new
HTTP/1.1 200 OK

[
{"SUCCESS": "Data source 'DS_old' renamed to 'DS_new'"}
]
```

404 Not Found - stated data source not found

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/DataSources/DS_old/DS_new
HTTP/1.1 404 Not Found

[
{"ERROR": "Data source 'DS_old' not found"}
]
```

400 Bad Request - old and new data source name not given

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/DataSources
```

```
HTTP/1.1 400 Bad Request

[
{"ERROR": "Old and/or new name of data source not provided"}
]
```

406 Not Acceptable - a data source with the new name already exists

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/DataSources/DS_old/DS_new
HTTP/1.1 406 Not Acceptable

[
{"ERROR": "There is already a data source bearing the name 'DS_new'"}
]
```

406 Not Acceptable - new data source name is invalid

For several reasons, the new data source name can be invalid:

Cause	Error Message
Starts with a period(`.`)	The new name must not begin with a period
Ends with a period (`.`)	The new name must not end with a period
Contains consecutive periods (`..`)	The new name must not contain a sequence of periods
Ends with `.TSF`	The new name must not end with '.TSF'

Output with one of the errors named above:

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/DataSources/MySQL2/MySQL.
HTTP/1.1 406 Not Acceptable

[
{"ERROR": "The new name must not end with a period."}
]
```

500 Internal Server Error - server side error

If a data source rename request was formally correct, but filed due to server side reasons, this output results, including a specific error message:

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/DataSources/MySQL1/MySQL2
HTTP/1.1 500 Internal Server Error

[
{"ERROR": "<error message>",
"SQL ErrorCode": "<error code>"
}
]
```

SAVEAS /DataSources → Save a copy of a data source

The method `SAVEAS` allows to create a copy of a data source. Therefore, the name and the new name of the data source need to be stated:

```
 curl -H "@auth" -iX SAVEAS http://127.0.0.1:8080/DataSources/DS_old/DS_new
HTTP/1.1 200 OK

[
{"SUCCESS": "Data source 'DS_old' saved as 'DS_new'"
}
]
```

The possible error messages are the same as for `RENAME`.

POST /DataSources → Add or replace a data source

The method `POST` allows to add a data source or change an existing one.

```
 curl -d @MySQL_DS -H "@auth" -iX POST http://127.0.0.1:8080/DataSources/
HTTP/1.1 201 Created

{"Success": "Data Source 'MySQL_new' created"}
```

The new data source is specified as a JSON object via data (`-d` in `curl`), for example as a text file like MySQL_DS in this call:

```
{
  "DataSources": [
    {
      "SourceName": "MySQL_new",
      "SourceType": "MYSQL",
      "Description": "",
      "AgentName": "Agent-Linux",
      "ConnectionString": ""SERVER=192.168.0.26;UID=user1;PWD=xxxxxxxxxxxxxxx"
    }
  ]
}
```

Note: If the data source is to be valid indefinitely, it is not necessary to specify `ValidFrom` or `ValidUntil`, the values "0000-00-00-00.00.00.000000" and "9999-99-99-99.99.99.999999" respectively are automatically set.

201 Created - New data source was added successfully

```
 curl -d @MySQL_DS -H "@auth" -iX POST http://127.0.0.1:8080/DataSources/
HTTP/1.1 201 Created

[
{"Success": "Data source 'MySQL_new' created"}
]
```

200 OK - Data source already existed

If the data source already existed, the output is as follows (regardless of whether a real change occurred or old and new data source were identical):

```
 curl -d @MySQL_DS -H "@auth" -iX POST http://127.0.0.1:8080/DataSources/
HTTP/1.1 200 OK

[
{"Success": "Data source 'MySQL_new' updated"}
]
```

400 Bad Request - no data source stated or not JSON compliant

No data source given:

```
 curl -H "@auth" -iX POST http://127.0.0.1:8080/DataSources/
HTTP/1.1 400 Bad Request

[
{"ERROR": "Request data missing"}
]
```

Data source not JSON compliant:

```
 curl -d @MySQL_DS -H "@auth" -iX POST http://127.0.0.1:8080/DataSources/
HTTP/1.1 400 Bad Request

[
{"ERROR": "Request data not JSON compliant"}
]
```

If the keywords `Name` or `SourceType` in the passed data are empty:

```
 curl -d @MySQL_DS -H "@auth" -iX POST http://127.0.0.1:8080/DataSources/
HTTP/1.1 400 Bad Request

[
{"ERROR": "Empty key fields"}
]
```

500 Internal Server Error - server side error

If a call changing an existing data source was formally correct, but failed due to server side reasons, this output results, including a specific error message:

```
 curl -d @MySQL_DS -H "@auth" -iX POST http://127.0.0.1:8080/DataSources/
HTTP/1.1 500 Internal Server Error

[
{"ERROR": "<error message>"}
]
```

PUT /DataSources → Change a data source

Contrary to `POST`, the use of `PUT` requires a data source as an argument, this is the one to be changed. In the JSON object, a data source name is not necessary, thus the keyword `Name` may be empty. If stated, it will be ignored.

As in `POST`, the data must be committed via header in JSON format.

```
 curl -d @MySQL_DS -H "@auth" -iX PUT http://127.0.0.1:8080/DataSources/MySQL_new
HTTP/1.1 200 OK

[
{"Success": "Data source 'MySQL_new' modified"}
]
```

400 Bad Request - data source not stated

If the name of the data source to be changed was not stated:

```
 curl -d @MySQL_DS -H "@auth" -iX PUT http://127.0.0.1:8080/DataSources/
HTTP/1.1 400 Bad Request

[
{"ERROR": "No argument with name of data source provided"}
]
```

If no data stated (for instance, file is empty or does not exist):

```
 curl -d @MySQ -H "@auth" -iX PUT http://127.0.0.1:8080/DataSources/SQL
HTTP/1.1 400 Bad Request

[
{"ERROR": "Empty request data"}
]
```

Note: If no data stated and no data source name given, the output is as in first example above.

If the data format is not JSON compliant (syntax errors, etc.):

```
 curl -d @MySQL1 -H "@auth" -iX PUT http://127.0.0.1:8080/DataSources/SQL
HTTP/1.1 400 Bad Request

[
{"ERROR": "Request data not JSON compliant"}
]
```

404 Not Found - stated data source in argument not found

Contrary to `POST`, `PUT` requires an existing data source. If the given data source does not exist:

```
 curl -d @MySQL_DS -H "@auth" -iX PUT http://127.0.0.1:8080/DataSources/SQL
HTTP/1.1 404 Not Found

[
{"ERROR": "Data source 'SQL' not found"}
]
```

500 Internal Server Error - server side error

If a call which was supposed to change an existing data source was formally correct, but failed due to server side reasons, this output results, including a specific error message:

```
 curl -d @MySQL_DS -H "@auth" -iX PUT http://127.0.0.1:8080/DataSources/MySQL_new
HTTP/1.1 500 Internal Server Error

{"Error": "<error message>",
"SQL ErrorCode: "<error code>"
}
```

DELETE /DataSources → Delete a data source

To delete a stated data source from the repository:

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/DataSources/MySQL_new
HTTP/1.1 200 OK

[
{"Success": "DataSource 'MySQL_new' deleted"}
]
```

400 Bad Request - no data source given

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/DataSources/
HTTP/1.1 400 Bad Request

[
{"ERROR": "Data source name not provided"}
]
```

404 Not Found - given data source does not exist

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/DataSources/MySQL_n
HTTP/1.1 404 Not Found

[
{"ERROR": "Data source 'MySQL_n' not found"}
]
```

500 Internal Server Error - server side error

If a data source delete request was formally correct, but failed due to server side reasons, this output results, including a specific error message:

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/DataSources/MySQL_new
HTTP/1.1 500 Internal Server Error

[
{"ERROR": "<error message>",
"SQL ErrorCode": "<error code>"}
]
```

Resource OutputTargets

The resource `OutputTargets` enables to print, create and modify output targets.

OPTIONS /OutputTargets → Help / method overview

```
 curl -H "@auth" -iX OPTIONS http://127.0.0.1:8080/OutputTargets
HTTP/1.1 200 OK

{"= HTTP method =                  ": "= Explanation =",
"GET    OutputTargets              ": "Print information on all Output Targets",
"GET    OutputTargets/<Name>       ": "Print information on an Output Target",
"RENAME OutputTargets/<Old>/<New> ": "Rename an Output Target"
"SAVEAS OutputTargets/<Old>/<New> ": "Save a copy of an Output Target"
"POST   OutputTargets              ": "Add a new or replace an Output Target",
"PUT    OutputTargets              ": "Change an Output Target",
"DELETE OutputTargets/<Name>       ": "Delete an Output Target",
"= Optional queries =              ": "= Explanation =",
"groupid=<GroupId>                   ": "Use <GroupId> (default value = 'DEFAULT')"}
```

GET /OutputTargets → List defined output targets

List all output targets:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputTargets
HTTP/1.1 200 OK

{"Results": [
{"OutputTargetName": "PG",
"GroupId": "DEFAULT",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"TargetType": "POSTGRESQL",
"TargetTypeLong": "Connection to PostgreSQL",
"Description": "",
"AgentName": "Agent_Linux",
"ConnectionString": "HOST=192.168.0.26;PORT=5432;DBNAME=postgres;USER=postgres;PASSWORD=xxxxxxxx
xxxxxxxxxxx;registerasreplayer=N",
"IdleTimeout": 0,
"OutputFlags": 40961,
"OutputFileMode": "WRITE",
"LoaderDecimalChar": ".",
"LBPSchema": "",
"Options": {
    "Use bound SQL-Parameter": true,
    "Bind parameters as array": true,
    "Send DML as block": false,
    "Generate multi-row-inserts in bulk processing": false,
    "Use quote identifiers": true,
    "Add CR/LF": false,
    "Generate comments": true,
    "Ignore DML errors": false,
    "Generate ignore insert error clause": false,
    "DDL changes: update output objects in repository": false,
    "Forward DDLs for captured input structure changes": false
}}]}
```

It is also possible to print a specific output target by stating its name (same status code as above):

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputTargets/PG
```

If no output targets are defined:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputTargets/
HTTP/1.1 200 OK

[
{"INFORMATION", "No defined output targets"}
]
```

404 Not Found - given output target does not exist

If the stated output target does not exist:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputTargets/Postg
HTTP/1.1 404 Not Found

[
{"ERROR": "Output target 'Postg' not found"}
]
```

Note: This output also results if no Output Targets are defined, but an Output Target was stated in the call.

RENAME OutputTargets → Rename an Output Target

Renaming an Output Target requires stating the old and new Output Target name:

```
curl -H "@auth" -iX RENAME http://127.0.0.1:8080/OutputTargets/OT_old/OT_new

HTTP/1.1 200 OK

{"Success": "Output Target 'OT_old' renamed to 'OT_new'"}
```

404 Not Found – stated Output Target not found

```
curl -H "@auth" -iX RENAME http://127.0.0.1:8080/OutputTargets/OT_old/OT_new
```

```
HTTP/1.1 404 Not Found

{"Error": "Output Target 'OT_old' not found"}
```

400 Bad Request – old and new name not given

```
curl -H "@auth" -iX RENAME http://127.0.0.1:8080/OutputTargets

HTTP/1.1 400 Bad Request

{"Error": "Old and/or new name of Output Target not provided"}
```

406 Not Acceptable – an Output Target with the new name already exists

```
curl -H "@auth" -iX RENAME http://127.0.0.1:8080/OutputTargets/OT_old/OT_new

HTTP/1.1 406 Not Acceptable

{"Error": "There is already an output bearing the name 'OT_new'"}
```

406 Not Acceptable – new Output Target name is invalid

For several reasons, the new Output Target name can be invalid.

Cause	Error Message
Starts with a period('.')	`The new name must not begin with a period`
Ends with a period ('.')	`The new name must not end with a period`
Contains consecutive periods ('..')	`The new name must not contain a sequence of periods`
Ends with '.TSF'	`The new name must not end with '.TSF'`

Output with one of the errors named above:

```
curl -H "@auth" -iX RENAME http://127.0.0.1:8080/OutputTargets/OTarget/OTarget.

HTTP/1.1 406 Not Acceptable

{"Error": "The new name must not end with a period."}
```

500 Internal Server Error – server side error

If an Output Target rename request was formally correct, but failed due to server side reasons, this output results, including a specific error message:

```
HTTP/1.1 500 Internal Server Error

{"Error": "<error message>",
"SQL ErrorCode": "<error code>"}
```

SAVEAS OutputTargets → Save a Copy of an Output Target

The method SAVEAS allows to create a copy of an Output Target. Therefore, the name and the new name of the Output Target need to be stated:

```
curl -H "@auth" -iX SAVEAS http://127.0.0.1:8080/OutputTargets/OT_old/OT_new

HTTP/1.1 200 OK

{"Success": "Output Target 'OT_old' saved as 'OT_new'"}
```

POST OutputTargets → Add or replace an Output Target

The method POST allows to add an output target or change an existing one.

```
 curl -d @PG.txt -H "@auth" -iX POST http://127.0.0.1:8080/OutputTargets/
HTTP/1.1 201 Created

{"Success": " Output Target 'PG' created"}
```

The new output target is specified as a JSON object via data (-d in curl), for example as a text file like PG.txt in this call:

```
{
  "OutputTargets": [
    {
      "OutputTargetName": "PG",
      "TargetType": "POSTGRESQL",
      "AgentName": "Agent-Linux",
      "ConnectionString": "HOST=localhost;PORT=5432;DBNAME=postgres;USER=postgres;PASSWORD=<pass
word>;registerasreplayer=N",
      "IdleTimeout": 8,
      "OutputFlags": 3187205,
      "OutputFileMode": "WRITE",
      "LoaderDecimalChar": "",
      "LBPSchema": ""
    }
  ]
}
```

Note: If the output target is to be valid indefinitely, it is not necessary to specify `ValidFrom` or `ValidUntil`, the values "0000-00-00-00.00.00.000000" and "9999-99-99-99.99.99.999999" respectively are automatically set.

201 Created - New output target was created successfully

```
 curl -H "@auth" -d @PG.txt -iX POST http://127.0.0.1:8080/OutputTargets
HTTP/1.1 201 Created

[
{"Success": "Output target 'PG' created"}
]
```

200 OK - Output target already existed

If the output target already existed, the output is as follows (regardless of whether a real change occurred or old and new output targets were identical):

```
 curl -H "@auth" -d @PG.txt -iX POST http://127.0.0.1:8080/OutputTargets
HTTP/1.1 200 OK

[
{"Success": "Output target 'PG' updated"}
]
```

PUT /OutputTargets → Change stated output target

Contrary to `POST`, the use of `PUT` requires an output target as an argument, this is the one to be changed. In the JSON object, an output target name is not necessary, thus the keyword `Name` may be empty. If stated, it will be ignored.

As in `POST`, the data must be committed via header in JSON format.

```
curl -d @PG.txt -H "@auth" -iX PUT http://127.0.0.1:8080/OutputTargets/PG
HTTP/1.1 200 OK

[
{"Success": "Output target 'PG' modified"}
]
```

400 Bad Request - output target not stated

If the name of the output target to be changed was not stated:

```
curl -d @PG.txt -H "@auth" -iX PUT http://127.0.0.1:8080/OutputTargets/
HTTP/1.1 400 Bad Request

[
{"Error":"No argument Output Target provided"}
]
```

If no data stated (for instance, file is empty):

```
curl -d @PG.txt -H "@auth" -iX PUT http://127.0.0.1:8080/OutputTargets/PG
HTTP/1.1 400 Bad Request

[
{"ERROR": "Empty request data"}
]
```

Note: If no data stated and no data source name given, the output is as in first example above.

If the data format is not JSON compliant (syntax errors, etc.):

```
curl -d @PG.txt -H "@auth" -iX PUT http://127.0.0.1:8080/OutputTargets/PG
HTTP/1.1 400 Bad Request

[
{"ERROR": "Request data not JSON compliant"}
]
```

404 Not Found - stated output target in argument not found

Contrary to POST, PUT requires an existing output target.

If the given output target does not exist:

```
curl -d @PG.txt -H "@auth" -iX PUT http://127.0.0.1:8080/OutputTargets/PG
HTTP/1.1 404 Not Found

[
{"ERROR": "Output Target 'PG' not found"}
]
```

500 Internal Server Error - server side error

If a call which was supposed to change an existing output target was formally correct, but failed due to server side reasons, this output results, including a specific error message:

```
curl -d @PG.txt -H "@auth" -iX PUT http://127.0.0.1:8080/OutputTargets/PG
HTTP/1.1 500 Internal Server Error

{"Error": "<error message>",
"SQL ErrorCode: "<error code>"
}
```

DELETE /OutputTargets → Delete stated output target

To delete a stated output target from the repository:

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/OutputTargets/PG
HTTP/1.1 200 OK

[
{"OutputTableSize": 0,
"ProcessDefsSize": 0,
"SUCCESS": "Output target 'PG' deleted"}
]
```

412 Precondition Failed - Output target is still referenced

If the output target is still referenced in output tables or process definitions, it can not be deleted. Therefore, it must be removed in the references first. The following output is generated, with indication of the references still present:

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/OutputTargets/PG
HTTP/1.1 412 Precondition Failed

{"Error": "Output Target 'PG' was NOT deleted",
"Information": "The Output Target 'PG' is still referenced in the following resource(s) (Type of
resource: [Name, ...])",
"Output tables": [
"MSSQL.DEMO.ARTICLE",
"MSSQL.PROD.ARTICLE"
],
"Process definitions": [
"BULK_MYSQL_2_MSSQL"]}
```

Resource Processes

The use of the resource `Processes` allows printing Rocket® Data Replicate and Sync process definitions, starting defined processes, listing running processes and stopping them as well as itemizing terminated processes. In addition, REXX variables of a running processes may be returned plus its latency.

Note: In Rocket® Data Replicate and Sync version 6, processes are called (processing) scripts.

OPTIONS /Processes -> Help / method overview

```
 curl -H "@auth" -iX OPTIONS http://127.0.0.1:8080/Processes
HTTP/1.1 200 OK


{"= HTTP method =                                    ":"= Explanation =",
"GET    Processes\/defined                           ":"Print definitions of all Processes",
"GET    Processes\/defined\/<Name>                   ":"Print definition of a Process",
"GET    Processes\/active                            ":"Print information on all running Process
es",
"GET    Processes\/active\/<Name>                    ":"Print information on one or more running
Processes",
"GET    Processes\/terminated                        ":"Print information on all terminated Proc
esses",
"GET    Processes\/terminated\/<Name>                ":"Print information on one or more termina
ted Processes",
"GET    Processes\/terminated\/<Name>\/<EndTime>     ":"Print information on a terminated Proces
s",
"GET    Processes\/    all                           ":"Print information on all Processes (acti
ve and terminated)",
"GET    Processes\/all\/<Name>                       ":"Print information on one or more Process
es (active and terminated)",
"GET    Processes\/all\/<Name>\/<EndTime>            ":"Print information on a Processes (active
and terminated)",
"GET    Processes\/rexxvar\/<Name>\/<PID>            ":"Print content of a REXX variable for an
active Process", "
   optional query: ?agent=<Agent>                   ":"   Use <Agent> (default value = '')",
"GET    Processes\/latency\/<Name>\/<PID>            ":"Print latency of an active Process",
"RENAME Processes\/defined\/<OldName>\/<NewName>     ":"Rename a Process definition",
"SAVEAS Processes\/defined\/<OldName>\/<NewName>     ":"Save a copy of a Process definition",
"POST   Processes\/defined                           ":"Add a Process definition",
"POST   Processes\/defined\/<Name>?action=start      ":"Start a Process definition",
"POST   Processes\/active\/<Name>\/<PID>?action=stop ":"Stop a running Processes",
"PUT    Processes\/defined                           ":"Change a Process definition",
"START  Processes\/defined\/<Name>                   ":"Start a Process definition",
"STOP   Processes\/active\/<Name>\/<PID>             ":"Stop a running Processes",
"DELETE Processes\/defined\/<Name>                   ":"Delete a Process definition",
"DELETE Processes\/terminated\/<Name>\/<EndTime>     ":"Delete a terminated Process (history ent
ry)",
"= Optional queries =                                ":"= Explanation =",
"groupid=<GroupId>                                   ":"Use <GroupId> (default value = 'DEFAUL
T')",
"starttime=<Timestamp>                                      ":"Use <Timestamp> when using GET active/te
rminated method"
"parameter=<Parameter>                                      ":"Use <Parameter> when using START metho
d"}
```

GET /Processes/defined -> List defined processes

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/defined
HTTP/1.1 200 OK

{"Results": [
{"GroupId":"DEFAULT",
"ProcessName": "CDC_PG2PG",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Parameters":
{"Input_Type": "CDC_NRT",
...
"Input_Source_Type": "POSTGRESQL"}},
{"GroupId":"DEFAULT",
"ProcessName": "DB2_2_PG_DEMO_ARTIKEL",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Parameters":
{"Input_Type": "BULK_TRANSFER",
...
"Input_Source_Type": "DB2"}},
{"GroupId":"DEFAULT",
"ProcessName": "PG2PG_new",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Parameters":
{"Input_Type": "BULK_TRANSFER",
...
"Input_Source_Type": "POSTGRESQL"}}
]}
```

It is also possible to print a specific process by stating its name (same status code as above):

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/defined/CDC_PG2PG
```

If no processes are defined:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/defined
HTTP/1.1 200 OK

[
{"INFORMATION": "No defined processes found."}
]
```

404 Not Found - given process definition does not exist

If the stated process definition does not exist:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/defined/CD
HTTP/1.1 404 Not Found

[
{"ERROR": "Stated process 'CD' not found"}
]
```

RENAME /Processes/defined -> Rename a defined process

Renaming a process definition requires stating the old and new process name:

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/Processes/defined/PG2PG/PG2PG_new
HTTP/1.1 200 OK

[
{"SUCCESS": "Process definition 'PG2PG' renamed to 'PG2PG_new'"}
]
```

404 Not Found - process definition not found

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/Processes/defined/PG2PG_/PG2PG_new
HTTP/1.1 404 Not Found

[
{"ERROR": "Process 'PG2PG_' not found"}
]
```

400 Bad Request - no new and old process name given

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/Processes/defined/PG2PG
HTTP/1.1 400 Bad Request

[
{"ERROR": "Old and/or new process name not provided"}
]
```

406 Not Acceptable - process is running

Renaming is not possible when there is a running process with the given name:

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/Processes/defined/DB2_2_PG_DEMO_ARTIKEL/DB2_P
G_DEMO
```

```
HTTP/1.1 406 Not Acceptable

[
{"ERROR": "The process 'DB2_2_PG_DEMO_ARTIKEL' is still running."}
]
```

406 Not Acceptable - A process definition with the new name exists already

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/Processes/defined/PG2PG/PG2PG_new
HTTP/1.1 406 Not Acceptable

[
{"ERROR": "There is already a process definition bearing the name 'PG2PG_new'."}
]
```

406 Not Acceptable - invalid new process name

For several reasons, the new process name can be invalid:

Cause	Error Message
Starts with a period(`.`)	The new name must not begin with a period
Ends with a period (`.`)	The new name must not end with a period
Contains consecutive periods (`..`)	The new name must not contain a sequence of periods
Ends with `.TSF`	The new name must not end with '.TSF'

Output with one of the errors named above:

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/Processes/defined/PG2PG/PG2PG.
HTTP/1.1 406 Not Acceptable

[
{"ERROR": "The new name must not end with a period."}
]
```

500 Internal Server Error - server side error

If a process rename request was formally correct, but failed due to server side reasons, this output results, including a specific error message:

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/Processes/defined/PG2PG/PG2PG_neu
HTTP/1.1 500 Internal Server Error

[
"ERROR": "<Error message",
"SQL ErrorCode": "<Error code>"}
```

```
]
```

SAVEAS Processes/defined -> Copy a defined Process

Copying a Process definition requires stating the original and new Process name:

```
curl -H "@auth" -iX SAVEAS http://127.0.0.1:8080/Processes/defined/PG2PG/PG2PG_copy
HTTP/1.1 200 OK

{"Success":"Process definition 'PG2PG' saved as 'PG2PG_copy'"}
```

404 Not Found – Process definition not found

```
curl -H "@auth" -iX SAVEAS http://127.0.0.1:8080/Processes/defined/PG2PG_/PG2PG_new
HTTP/1.1 404 Not Found

{"Error": "Process 'PG2PG_' not found"}
```

400 BadRequest – no original and new Process name given

```
curl -H "@auth" -iX SAVEAS http://127.0.0.1:8080/Processes/defined/PG2PG
HTTP/1.1 400 Bad Request

{"Error": "Process name and/or new Process name not provided"
```

406 Not Acceptable – A Process definition with the new name exists already

```
curl -H "@auth" -iX SAVEAS http://127.0.0.1:8080/Processes/defined/PG2PG/PG2PG_copy
HTTP/1.1 406 Not Acceptable

{"Error": "There is already a Process definition bearing the name 'PG2PG_copy'."}
```

406 Not Acceptable – invalid new Process name

For several reasons, the new Process name can be invalid:

Cause	Error Message
Starts with a period('.')	The new name must not begin with a period
Ends with a period ('.')	The new name must not end with a period
Contains consecutive periods ('..')	The new name must not contain a sequence of periods
Ends with '.TSF'	The new name must not end with '.TSF'

Output with *one* of the errors named above:

```
curl -H "@auth" -iX SAVEAS http://127.0.0.1:8080/Processes/defined/PG2PG/PG2PG.
HTTP/1.1 406 Not Acceptable

{"Error": "The new name must not end with a period."}
```

500 Internal Server Error – server side error

If a Process copy request was formally correct, but failed due to server side reasons, this output results, including a specific error message:

```
curl -H "@auth" -iX SAVEAS http://127.0.0.1:8080/Processes/defined/PG2PG/PG2PG_copy
HTTP/1.1 500 Internal Server Error

"Error": "<Error message",
"SQL ErrorCode": "<Error code>"}
```

POST Processes/defined -> Insert process definitions

The call allows process definitions to be inserted into the repository.

For the insert all field values of the objects must be stored in a file in JSON format, and this file is then sent in the request data. The names of the field values correspond to the field names in repository table `ProcessDefs`, and are not case sensitive.

The individual process definition objects are embedded within the JSON array `ProcessDefs`, this identifier is also not case sensitive.

Note: Process definitions have a variety of parameters. These are defined within the JSON object “Parameters” which is embedded in the process object.

Two process definitions are to be inserted into the repository table `ProcessDefs`. To do this, a JSON file `data.json` with the following content (shown in abbreviated form) is created.

```
{
"ProcessDefs": [
{"GroupId":"DEFAULT",
"ProcessName":"db2tokafka",
"ValidFrom":"0000-00-00-00.00.00.000000",
"ValidUntil":"9999-99-99-99.99.99.999999",
"Parameters": {"Input_Type": "BULK_TRANSFER",
"Input_Source_Name": "DB9D",
"Function": "CDC",
...
"Input_Source_Query_Table": "DB9D.DEMO.*"}
},
{"GroupId":"DEFAULT",
"ProcessName":"db2topostgres",
"ValidFrom":"0000-00-00-00.00.00.000000",
"ValidUntil":"9999-99-99-99.99.99.999999",
"Parameters": {"Input_Type": "BULK_TRANSFER",
"Input_Source_Name": "DB8D",
"Function": "CDC",
...
"Input_Source_Query_Table":"DB8D.TEST.*"}
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/Processes/defined"
HTTP/1.1 200 OK

{"ProcessDefs inserted":[
{"GroupId":"DEFAULT","ProcessName":"db2tokafka","ValidFrom":"0000-00-00-00.00.00.000000","ValidU
ntil":"9999-99-99-99.99.99.999999"},
{"GroupId":"DEFAULT","ProcessName":"db2topostgres","ValidFrom":"0000-00-00-00.00.00.000000","Val
idUntil":"9999-99-99-99.99.99.999999"}
]}
```

500 Internal Server Error – server side error

If an insert request was formally correct, but failed due to server side reasons, this output results, including a specific error message:

```
HTTP/1.1 500 Internal Server Error

{"Error":"Processes failed",
"Reason":"Got SQL error message while inserting ",
"Object":"ProcessDef DEFAULT.db2tokafka (0000-00-00-00.00.00.000000 - 9999-99-99-99.99.99.99999
9)","SQL message":"TCS0597E;70R1940,WIN64_6.2.9200; Repository SQL error: TCS0256E;70R1940,WIN6
4_6.2.9200; PostgreSQL libpq SQLSTATE 23505, ERROR: duplicate key value violates unique constrai
nt »processdefs_pkey« DETAIL: key »(groupid, processname, validfrom, parametername)=(DEFAULT , d
b2tokafka , 0000-00-00-00.00.00.000000, Input_Source_Name )« already exists.; N\/A."}curl -d @da
ta.json -H "@auth" -iX PUT "http://127.0.0.1:8080/Processes/defined"
```

400 Bad Request – Not all key values provided

If not all key values are provided to uniquely identify an object, this will be indicated with a message like this.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/Processes/defined"
HTTP/1.1 400 Bad Request

{"Error":"Processes failed",
"Reason":"Inserting 'ProcessDefs' object in repository not possible, because not all key values
have been specified.",
"Object":"ProcessDef DEFAULT. (0000-00-00-00.00.00.000000 - 9999-99-99-99.99.99.999999)"}
```

UPDATE Processes/defined -> Update Process definitions

The call allows process definitions to be updated in the repository.

For the update all field values of the objects must be stored in a file in JSON format, and this file is then sent in the request data. The names of the field values correspond to the field names in repository table `ProcessDefs`, and are not case sensitive. The names of the key values of the object to be updated is preceded by the prefix `Before_`.

The individual process definition objects are embedded within the JSON array `ProcessDefs`, this identifier is also not case sensitive.

Note: Process definitions have a variety of parameters. These are defined within the JSON array “Parameters” which is embedded in the process object. The following applies:

- A parameter present in the original process definition is deleted if it does not exist within the defined JSON array "Parameters".
- A parameter present in the original process definition is updated if it also exists within the defined JSON array "Parameters".
- A parameter not present in the original process definition is added if it exists within the defined JSON array "Parameters".

A process definition is to be inserted into the repository table `ProcessDefs`. To do this, a JSON file `data.json` with the following content (shown in abbreviated form) is created.

```
{
"ProcessDefs": [
{"Before_GroupId":"DEFAULT",
"Before_ProcessName":"db2tokafka",
"Before_ValidFrom":"0000-00-00-00.00.00.000000",
"Before_ValidUntil":"999999-99-99.99.99.999999",
"GroupId":"DEFAULT",
"ProcessName":"db2tokafka",
"ValidFrom":"0000-00-00-00.00.00.000000",
"ValidUntil":"9999-99-99-99.99.99.999999",
"Parameters":[
{"Input_Type": "BULK_TRANSFER",
"Input_Source_Name": "DBED",
"Function": "CDC",
...
"Input_Source_Query_Table":"DBED.DEMO.*"}
] } ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX UPDATE "http://127.0.0.1:8080/Processes/defined"
HTTP/1.1 200 OK

{"ProcessDefs updated":[
{"GroupId":"DEFAULT","ProcessName":"db2tokafka","ValidFrom":"0000-00-00-00.00.00.000000","ValidU
ntil":"999999-99-99.99.99.999999"}
]}
```

400 Bad Request – Not all key values provided

If not all key values are provided to uniquely identify an object, this will be indicated with a message like this:

```
curl -d @data.json -H "@auth" -iX UPDATE "http://127.0.0.1:8080/Processes/defined"
HTTP/1.1 400 Bad Request

{"Error":"Processes failed",
"Reason":"Updating 'ProcessDefs' object in repository not possible, because not all key values o
f 'Before' object have been specified.",
"Object":"ProcessDef DEFAULT. (0000-00-00-00.00.00.000000 - 999999-99-99.99.99.999999)"}
```

400 Bad Request – Process definition not found

If no object matching the key values passed is found in the repository, this will be indicated with a message like this:

```
curl -d @data.json -H "@auth" -iX UPDATE "http://127.0.0.1:8080/Processes/defined"
HTTP/1.1 400 Bad Request

{"Error":"Processes failed",
"Reason":"Updating 'ProcessDefs' object in repository not possible, because object does not exis
t in repository.",
"Object":"ProcessDef DEFAULT.db2tokafka (0000-00-00-00.00.00.000000 - 999999-99-99.99.99.99999
9)"}
```

START /Processes/defined -> Start defined process

To start a process, the name of the process definition must be specified. The process ID of the process started in this way is also appended to the returned JSON object:

```
 curl -H "@auth" -iX START http://127.0.0.1:8080/Processes/defined/DB2_2_PG_DEMO_ARTIKEL
HTTP/1.1 200 OK

[
{"SUCCESS": "The process 'DB2_2_PG_DEMO_ARTIKEL' has been successfully started on agent Agent_Li
nux",
"ProcessID": "000EC1B1",
"ParameterString": ""
]
```

It is also possible to state one or more parameters as a string with key-value pair(s). This is done via the query `Parameter=<Parameter(s)>` . Multiple key-value pairs are separated by blanks, denoted as `%20`. If a value itself contains blanks, it must be surrounded by single or double quotes, encoded as `%27` (`'`) or `%22` (`"`), respectively. The parameter string is printed in the JSON output. If double quotes are used, they are shown as `\"`, since `"` already encloses strings.

Example for a URI with parameter string `PM_I.INPUT_SOURCE_NAME=ARTICLE.DAT`:

`http://127.0.0.1:8080/Processes/defined/`
`DB2_2_PG_DEMO_ARTIKEL?Parameter=PM_I.INPUT_SOURCE_NAME=ARTICLE.DAT`

Example for a URI with parameter string `PAR1=1 PAR2='abc def'`:

`http://127.0.0.1:8080/Processes/defined/`
`DB2_2_PG_DEMO_ARTIKEL?Parameter=PAR1=1%20PAR2=%27abc%20def%27`

404 Not Found - process definition does not exist

```
 curl -H "@auth" -iX START http://127.0.0.1:8080/Processes/defined/PG
HTTP/1.1 404 Not Found

[
{"ERROR": "Process 'PG' not found"}
]
```

404 Not Found - missing agent to start onto

In the process definition, the property `Start_Process_On` must specify on which agent the process is to run (usually correspond to the input agent of the process). If this specification is omitted, the output is the following:

```
 curl -H "@auth" -iX START http://127.0.0.1:8080/Processes/defined/DB2_2_PG_DEMO_ARTIKEL
HTTP/1.1 404 Not Found

[
{"ERROR": "Have no agent to start onto."}
]
```

400 Bad Request - name of defined process not given

```
 curl -H "@auth" -iX START http://127.0.0.1:8080/Processes/defined/
HTTP/1.1 400 Bad Request

[
{"ERROR": "No process stated"}
]
```

406 Not Acceptable - process could not be started

```
 curl -H "@auth" -iX START http://127.0.0.1:8080/Processes/defined/DB2_2_PG_DEMO_ARTIKEL
HTTP/1.1 406 Not Acceptable

[
{"ERROR": "The process 'DB2_2_PG_DEMO_ARTIKEL' could not be started on agent Agent_Linux"}
]
```

POST /Processes/defined and action=start -> Start defined process

To start a process, the name of the process definition must be specified.

The process ID of the process started in this way is also appended to the returned JSON object:

```
 curl -H "@auth" -iX POST http://127.0.0.1:8080/Processes/defined/DB2_2_PG_DEMO_ARTIKEL?action=s
```

```
tart
HTTP/1.1 200 OK

[
{"SUCCESS": "The process 'DB2_2_PG_DEMO_ARTIKEL' has been successfully started on agent Agent_Li
nux",
"ProcessID": "000EC1B1",
"ParameterString": ""
]
```

It is also possible to state one or more parameters as a string with key-value pair(s). This is done via the query **Parameter=<Parameter(s)>**.

Multiple key-value pairs are separated by blanks, denoted as %20. If a value itself contains blanks, it must be surrounded by single or double quotes, encoded as %27 (') or %22 ("), respectively. The parameter string is printed in the JSON output. If double quotes are used, they are shown as `\"`, since `"` already encloses strings.

Example for a URI with parameter string `PM_I.INPUT_SOURCE_NAME=ARTICLE.DAT`:

http://127.0.0.1:8080/Processes/defined/
DB2_2_PG_DEMO_ARTIKEL?Parameter=PM_I.INPUT_SOURCE_NAME=ARTICLE.DAT

Example for a URI with parameter string `PAR1=1 PAR2='abc def'`:

http://127.0.0.1:8080/Processes/defined/
DB2_2_PG_DEMO_ARTIKEL?Parameter=PAR1=1%20PAR2=%27abc%20def%27

404 Not Found - process definition does not exist

```
 curl -H "@auth" -iX POST http://127.0.0.1:8080/Processes/defined/PG?action=start
HTTP/1.1 404 Not Found

[
{"ERROR": "Process 'PG' not found"}
]
```

404 Not Found - missing agent to start onto

In the process definition, the property `Start_Process_On` must specify on which agent the process is to run (usually correspond to the input agent of the process).

If this specification is omitted, the output is the following:

```
 curl -H "@auth" -iX POST http://127.0.0.1:8080/Processes/defined/DB2_2_PG_DEMO_ARTIKEL?action=s
tart
HTTP/1.1 404 Not Found

[
{"ERROR": "Have no agent to start onto."}
]
```

400 Bad Request - name of defined process not given

```
 curl -H "@auth" -iX POST http://127.0.0.1:8080/Processes/defined/?action=start
HTTP/1.1 400 Bad Request

[
{"ERROR": "No process stated"}
]
```

406 Not Acceptable - process could not be started

```
 curl -H "@auth" -iX POST http://127.0.0.1:8080/Processes/defined/DB2_2_PG_DEMO_ARTIKEL?action=s
tart
HTTP/1.1 406 Not Acceptable

[
{"ERROR": "The process 'DB2_2_PG_DEMO_ARTIKEL' could not be started on agent Agent_Linux"}
]
```

GET /Processes/active -> Print running processes

To list all running process:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/active
HTTP/1.1 200 OK

{"Active Processes": true,
"Results": [
{"ProcessName": "DB2_2_PG_DEMO_ARTIKEL",
"AgentName": "Agent_Linux",
"ProcessType": "Data Process",
"StartTime": "2022-07-26-14.08.39.817755",
"EndTime": "9999-99-99-99.99.99.999999",
"ActiveFor": "23 hours, 58 minutes, 51 seconds",
"ProcessState": "RUNNING",
"ReturnCode": "",
"ErrorCode": 0}]}
```

To print details of an active process and also of process parts, state the process name as a parameter:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/active/DB2_2_PG_DEMO_ARTIKEL
HTTP/1.1 200 OK

{"Results": [
{"ProcessName": "DB2_2_PG_DEMO_ARTIKEL:Input:Processing",
"AgentName": "Agent_Linux",
"ProcessType": "Data Process",
"StartTime": "2022-07-26-14.08.39.817755",
"EndTime": "9999-99-99-99.99.99.999999",
"ActiveFor": "1 day, 0 hours, 1 minute, 9 seconds",
"ProcessState": "RUNNING",
"ReturnCode": "",
"ErrorCode": 0,
"ProcessPartStart": "2022-07-26-14.08.39.805616",
"LastUpdated": "2022-07-26-22.00.05.590141",
"LastProcessTst": "2022-07-26-21.46.13.781856",
"ProcessPid": "001ACF0C",
"RemoteAgentName": "",
"RemoteProcessName": "",
"RemoteProcessPid": "",
"ProtocolFileFlag": 8,
"ControlRecsInput": 0,
"DataBlocksInput": 6582,
"DataRecsInput": 17,
"ControlRecsOutput": 0,
"DataBlocksOutput": 0,
"DataRecsOutput": 0,
"LUWCount": 1,
"LUWMaxConcurrent": 1,
"LUWUncommittedCount": 0,
"BUBCInsertCount": 0,
"BUBCDeleteCount": 0,
"BUBCUpdateCount": 0,
"BUBCUnchangedCount": 0},
{"ProcessName": "DB2_2_PG_DEMO_ARTIKEL:Processing:Applying",
"AgentName": "Agent_Linux",
"ProcessType": "Data Process",
"StartTime": "2022-07-26-14.08.39.817755",
"EndTime": "9999-99-99-99.99.99.999999",
"ActiveFor": "1 day, 0 hours, 1 minute, 9 seconds",
"ProcessState": "RUNNING",
"ReturnCode": "",
"ErrorCode": 0,
"ProcessPartStart": "2022-07-26-14.08.39.964428",
"LastUpdated": "2022-07-26-22.00.05.644500",
"LastProcessTst": "",
"ProcessPid": "001ACF39",
"RemoteAgentName": "",
"RemoteProcessName": "",
"RemoteProcessPid": "",
"ProtocolFileFlag": 8,
"ControlRecsInput": 0,
"DataBlocksInput": 0,
"DataRecsInput": 0,
"ControlRecsOutput": 0,
"DataBlocksOutput": 0,
"DataRecsOutput": 0,
"LUWCount": 0,
"LUWMaxConcurrent": 0,
"LUWUncommittedCount": 0,
"BUBCInsertCount": 0,
"BUBCDeleteCount": 0,
"BUBCUpdateCount": 0,
"BUBCUnchangedCount": 0}]}
```

It is also possible to only display a sub-process (output as above, but only containing the sub-process):

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/active/
DB2_2_PG_DEMO_ARTIKEL:Processing:Applying
```

If there are no active processes meeting the search criteria the status code `204` is returned.

GET /Processes/terminated -> Print terminated processes

To list all terminated process runs:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/terminated
HTTP/1.1 200 OK

{"Results": [
{"ProcessName": "PG2PG",
"AgentName": "Agent_Linux",
"ProcessType": "Data Process",
"StartTime": "2021-10-08-08.33.48.294807",
"EndTime": "2021-10-08-08.36.35.261210",
"ProcessState": "TERMINATED",
"ReturnMessage": "",
"Duration": "2 minutes, 47 seconds",
"ReturnCode": "00000000",
"ErrorCode": 0},
{"ProcessName": "DB2_2_PG_DEMO_ARTIKEL",
"AgentName": "Agent_Linux",
"ProcessType": "Data Process",
"StartTime": "2022-07-26-12.53.59.436647",
"EndTime": "2022-07-26-14.05.51.125783",
"ProcessState": "TERMINATED",
"ReturnMessage": "",
"Duration": "1 hour, 11 minutes, 52 seconds",
"ReturnCode": "00000000",
"ErrorCode": 0}]}
```

Likewise, a detailed output results, if a process name is given as a parameter.

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/terminated/PG2PG
HTTP/1.1 200 OK

{"Results": [
{"ProcessName": "PG2PG:Input:Processing:Applying",
"AgentName": "Agent_Linux",
"ProcessType": "Data Process",
"StartTime": "2021-10-08-08.33.48.294807",
"EndTime": "2021-10-08-08.36.35.261210",
"ProcessState": "TERMINATED",
"ReturnMessage": "",
"Duration": "2 minutes, 47 seconds",
"ReturnCode": "00000000",
"ErrorCode": 0,
"ProcessPartStart": "2021-10-08-08.33.48.276914",
"LastUpdated": "2021-10-08-08.36.35.327586",
"LastProcessTst": "2021-10-08-08.36.35.187723",
"ProcessPid": "00000000",
"RemoteAgentName": "",
"RemoteProcessName": "",
"RemoteProcessPid": "",
"ProtocolFileFlag": 0,
"ControlRecsInput": 0,
"DataBlocksInput": 4370,
"DataRecsInput": 1002040,
"ControlRecsOutput": 2,
"DataBlocksOutput": 0,
"DataRecsOutput": 1002040,
"LUWCount": 0,
"LUWMaxConcurrent": 0,
"LUWUncommittedCount": 0,
"BUBCInsertCount": 0,
"BUBCDeleteCount": 0,
"BUBCUpdateCount": 0,
"BUBCUnchangedCount": 0}]}
```

Specification of an `EndTime`: Since there can be several terminated processes with the same name, it is possible to specify the end time of the desired process run (`EndTime`) as an argument in order to display only this individual process run:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/all/DB2_2_PG_DEMO_ARTIKEL/
2022-07-26-14.05.51.125783
```

GET /Processes/all -> Print running and terminated processes

To display active and terminated process runs, there is the context `/Processes/all`. Process definitions are not included (`/Processes/defined`).

If no process name is stated, it is additionally displayed whether there are running processes or not.

```
{"Active processes": "true"}
```

or

```
{"Active processes": "false"}
```

Apart from that, the results are equal to `Processes/terminated` or `Processes/active`. It is also possible to specify `Process` and `EndTime` here.

GET /Processes/rexxvar -> REXX variable readout

In order to output a REXX variable for a runnung process, the name of the REXX variable is required in addition to `Process` and `ProcessID`:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/rexxvar/DB2_2_PG_DEMO_ARTIKEL/001ACF39/
PM_O.LastProcessTimestamp
HTTP/1.1 200 OK

{"RexxVariable": "PM_O.LastProcessTimestamp",
"RexxVariableContent": "2022-07-28-11.57.53.992905",
"ProcessName": "DB2_2_PG_DEMO_ARTIKEL:Processing:Applying",
"ProcessPid": "00002644",
"AgentName": "Agent-Windows",
"RemoteAgentName": "",
"RemoteProcessName": "",
"RemoteProcessPid": ""}
```

400 Bad Request - Process and/or ProcessID not given

Both `Process` and `ProcessID` need to be stated. If one of them or both are missing, this output results:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/rexxvar/DB2_2_PG_DEMO_ARTIKEL/001AC/P
M_O.LastProcessTimestamp
HTTP/1.1 400 Bad Request

[
{"ERROR": "Process and/or ProcessID not stated"}
]
```

500 Internal Server Error - server side error while trying to read out a REXX variable

If a readout request of a process' REXX variable was formally correct, but failed due to server side reasons, this output results, including a specific error message:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/rexxvar/DB2_2_PG_DEMO_ARTIKEL/001ACF39/
PM_O.LastPr
HTTP/1.1 500 Internal Server Error

[
{"RexxVariable": "PM_O.LastPr",
"ERROR": "Agent process communication: REXX parameter unknown.",
"ErrorCode": "0E0000A2"}
]
```

GET /Processes/latency -> Report latency

To print the latency of a running process, stating `Process` and `ProcessID` is vital:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/latency/DB2_2_PG_DEMO_ARTIKEL/001ACF39
HTTP/1.1 200 OK

{"Latency_Milliseconds": 28660,
"Latency_HumanReadable": "29 seconds",
"UsedProcessPart": "DB2_2_PG_DEMO_ARTIKEL:Processing:Applying",
"ProcessPid": "00002644",
"AgentName": "Agent-Windows",
"RemoteAgentName": "",
"RemoteProcessName": "",
"RemoteProcessPid": ""}
```

500 Internal Server Error - server side error while determining latency

If requesting a process' latency was formally correct, but failed due to server side reasons, this output results, including a specific error message:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/latency/DB2_2_PG_DEMO_ARTIKEL/001ACF39
HTTP/1.1 500 Internal Server Error

[
"ERROR": "<error message>",
"SQL ErrorCode": "<error code>"}
]
```

STOP /Processes/active -> Stop a running process

In order to stop a running process, the process ID must also be specified in addition to the process name for reasons of clarity. The `ProcessID` is reported after starting the process. The `ProcessId` or the `StartTime` can also be determined by `GET /Processes/active/<Process>` , see "GET /Processes/active → Print running processes".

```
 curl -H "@auth" -iX STOP http://127.0.0.1:8080/Processes/active/DB2_2_PG_DEMO_ARTIKEL/000EC1B1
HTTP/1.1 200 OK


[
{"SUCCESS": "The process DB2_2_PG_DEMO_ARTIKEL with ProcessID '000EC1B1' has been successfully s
topped on agent Agent_Linux"}
]
```

Or,

```
 curl -H "@auth" -iX STOP http://127.0.0.1:8080/Processes/active/DB2_2_PG_DEMO_ARTIKEL/?starttim
e=2024-11-14-12.33.48.885535
HTTP/1.1 200 OK
[
{"SUCCESS": "The process DB2_2_PG_DEMO_ARTIKEL with ProcessID '000EC1B1' has been successfully s
topped on agent Agent_Linux"}
]
```

404 Not Found - process does not exist

If no process with the given `ProcessID` could be found, or there is no running process with the stated `Process`, or the process is already terminated:

```
 curl -H "@auth" -iX STOP http://127.0.0.1:8080/Processes/active/DB2_2_PG_DEMO_ARTIKEL/000472FC
HTTP/1.1 404 Not Found
```

```
[
{"ERROR": "The process 'DB2_2_PG_DEMO_ARTIKEL' with ProcessID '000472FC' could not be found or i
s already terminated."}
]
```

400 Bad Request - Process or ProcessID not given

If `Process` or `ProcessID` was missing during the call:

```
 curl -H "@auth" -iX STOP http://127.0.0.1:8080/Processes/active/DB2_2_PG_DEMO_ARTIKEL
HTTP/1.1 400 Bad Request

[
{"ERROR": "Process not stated"}
]
```

Or similar, depending on the reason.

500 Internal Server Error - server side error while trying to stop a process

If a process stop request was formally correct, but failed due to server side reasons, this output results, including a specific error message:

```
 curl -H "@auth" -iX STOP http://127.0.0.1:8080/Processes/CDC_PG2PG/003F8F69
HTTP/1.1 500 Internal Server Error

[
{"ERROR": "The process 'DC_PG2PG' with ProcessID '003F8F69' could not be stopped on agent Agen
t_Linux"}
]
```

POST /Processes/active and action=stop -> Stop a running process

In order to stop a running process, the process ID or the start time must also be specified in addition to the process name for reasons of clarity.

The `ProcessID` is reported after starting the process. The `ProcessId` or the `StartTime` can also be determined by **`GET /Processes/active/<Process>`** , see "GET /Processes/active → Print running processes".

```
 curl -H "@auth" -iX POST http://127.0.0.1:8080/Processes/active/DB2_2_PG_DEMO_ARTIKEL/000EC1B
1&action=stop


HTTP/1.1 200 OK
[
{"SUCCESS": "The process DB2_2_PG_DEMO_ARTIKEL with ProcessID '000EC1B1' has been successfully s
topped on agent Agent_Linux"}
]
```

Or,

```
 curl -H "@auth" -iX POST http://127.0.0.1:8080/Processes/active/DB2_2_PG_DEMO_ARTIKEL?starttim
e=2024-11-14-12.33.48.885535&action=stop

HTTP/1.1 200 OK
[
{"SUCCESS": "The process DB2_2_PG_DEMO_ARTIKEL with ProcessID '000EC1B1' has been successfully s
topped on agent Agent_Linux"}
]
```

404 Not Found - process does not exist

If no process with the given `ProcessID` could be found, or there is no running process with the stated Process, or the process is already terminated:

```
 curl -H "@auth" -iX POST http://127.0.0.1:8080/Processes/active/DB2_2_PG_DEMO_ARTIKEL/000472F
C&action=stop


HTTP/1.1 404 Not Found
[
{"ERROR": "The process 'DB2_2_PG_DEMO_ARTIKEL' with ProcessID '000472FC' could not be found or i
s already terminated."}
]
```

400 Bad Request - Process or ProcessID not given

If `Process` or `ProcessID` was missing during the call:

```
 curl -H "@auth" -iX POST http://127.0.0.1:8080/Processes/active/DB2_2_PG_DEMO_ARTIKEL&action=st
op
HTTP/1.1 400 Bad Request

[
{"ERROR": "Process not stated"}
]
```

Or similar, depending on the reason.

500 Internal Server Error - server side error while trying to stop a process

If a process stop request was formally correct, but failed due to server side reasons, this output results, including a specific error message:

```
 curl -H "@auth" -iX POST http://127.0.0.1:8080/Processes/CDC_PG2PG/003F8F69&action=stop
HTTP/1.1 500 Internal Server Error

[
{"ERROR": "The process 'DC_PG2PG' with ProcessID '003F8F69' could not be stopped on agent Agen
t_Linux"}
]
```

DELETE /Processes/defined -> Delete defined process

To delete a process definition:

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/Processes/defined/PG2PG
HTTP/1.1 200 OK

[
{"SUCCESS": "Process definition 'PG2PG' deleted"}
]
```

404 Not found - The given process was not found

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/Processes/defined/PG2P
HTTP/1.1 404 Not Found

[
{"ERROR": "Process 'PG2P' not found"}
]
```

406 Not Acceptable - A process with this process definition is still active

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/Processes/defined/DB2_2_PG_DEMO_ARTIKEL
HTTP/1.1 406 Not Acceptable

[
{"ERROR": "The process 'DB2_2_PG_DEMO_ARTIKEL' is still running."}
]
```

406 Not Acceptable - The process is still referenced

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/Processes/defined/DB2_2_PG_DEMO_ARTIKEL
HTTP/1.1 406 Not Acceptable

[
{"ERROR": The process 'DB2_2_PG_DEMO_ARTIKEL' is still referenced in: 'DB2', job 'job1' as proce
ss"}
]
```

DELETE /Processes/terminated -> Delete a terminated process

To delete a terminated process from the process history:

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/Processes/terminated/DB2_2_PG/2022-10-25-13.3
7.19.213788
HTTP/1.1 200 OK

[
{"SUCCESS": "Deleted process(es):'DB2_2_PG:Input:Processing' (EndTime:2022-10-25-13.37.19.21378
8), 'DB2_2_PG:Processing:Applying' (EndTime:2022-10-25-13.37.19.230294)"}
]
```

404 Not Found - No process with given name or given EndTime

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/Processes/terminated/DB2_2_PG/2022-10-25-13.3
7.19.213788
HTTP/1.1 404 Not Found

[
{"ERROR": "Stated process 'DB2_2_PG' with EndTime '2022-10-25-13.37.19.213788' not found."}
]
```

400 Bad Request - No EndTime stated

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/Processes/terminated/DB2_2_PG
HTTP/1.1 400 Bad Request

[
{"ERROR": "EndTime of process not stated."}
]
```

406 Not Acceptable - EndTime 9999-99-99-99.99.99.999999 given

Processes with an Endtime of 9999-99-99-99.99.99.999999 are running processes and thus can not be deleted from the process history.

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/Processes/terminated/DB2_2_PG/9999-99-99-99.9
9.99.999999
HTTP/1.1 406 Not Acceptable

[
{"ERROR": "Deletion of a running process is not possible"}
]
```

Resource Jobs (Scheduler)

Scheduling is used to handle jobs. A job contains one or more processes and information about the timing of these processes. The creation of jobs can be as complex as you like and is therefore only possible with the Rocket® Data Replicate and Sync dashboard. With the REST API, jobs created in this way can be displayed, renamed, activated, and deleted.

OPTIONS /Jobs -> Method overview

```
 curl -H "@auth" -iX OPTIONS http://127.0.0.1:8080/Jobs
HTTP/1.1 200 OK

{"= HTTP method =                      ": "= Explanation =",
"GET    Jobs/defined                  ": "List defined Scheduler Jobs",
"RENAME Jobs/defined/<Name/<NewName>  ": "Rename a Job",
"SAVEAS Jobs/defined/<Name/<NewName>  ": "Save a copy of a Job",
"DELETE Jobs/defined/<Name>           ": "Delete a defined Scheduler Job",
"START  Jobs/defined/<Name>           ": "Schedule a Job",
"GET    Jobs/events/<Name>            ": "List Events of a Job",
"STOP   Jobs/events/<Name>/<EventID>  ": "Unschedule a Job Event",
"DELETE Jobs/events/<Name>/<EventID>  ": "Delete a Job Event (list  entry)",
"= Optional queries =                 ": "= Explanation =",
"groupid=<GroupId>                      ": "Use <GroupId> (default value = 'DEFAULT')"}
```

GET /Jobs -> Status information

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Jobs
HTTP/1.1 200 OK

[
{"JobName": "Job1",
"JobDescription": "",
"GroupID": "DEFAULT",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"JobSteps": 3,
"JobStep 1": "Type=Condition;Start=Y;Name=Jobstart;StartAgent=Agent_Linux;DeleteScriptsFromHisto
ry=Y;Gen=1;OUT_00000=T00001",
"JobStep 2": "Type=Process;Name=PG2PG;ContinueType=1;ContinueIn=17.00.00.000000;Gen=2",
"JobStep 3": "Type=Condition;Name=JobEnd - checking PG2PG;Gen=3;IN_00000=T00001"}
]
```

It is also possible to print a specific job by stating its name (same status code as above):

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/Jobs/Job1
```

If no jobs are defined:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Jobs
```

```
HTTP/1.1 200 OK

[
{"INFORMATION", "No defined jobs found."}
]
```

404 Not Found - stated job does not exist

If the stated job does not exist:

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Jobs/Job2
HTTP/1.1 404 Not Found

[
{"ERROR": "Stated job 'Job2' not found."}
]
```

RENAME /Jobs -> Rename a job

Renaming a job requires stating the old and new job name:

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/Jobs/Job_new/Job2
HTTP/1.1 200 OK

[
{"SUCCESS": "Job 'Job_new' renamed to 'Job2'"}
]
```

404 Not Found - stated job not found

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/Jobs/Jobnew/Job2
HTTP/1.1 404 Not Found

[
{"ERROR": "Job 'Jobnew' not found"}
]
```

404 Bad Request - old and new job name not given

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/Jobs/Job_new
HTTP/1.1 400 Bad Request

[
{"ERROR": "Old and/or new name of job not provided"}
]
```

406 Not Acceptable - invalid new job name

For several reasons, the job name can be invalid:

Cause	Error Message
Starts with a period(`.`)	The new name must not begin with a period
Ends with a period (`.`)	The new name must not end with a period
Contains consecutive periods (`..`)	The new name must not contain a sequence of periods
Ends with `.TSF`	The new name must not end with '.TSF'

Output with one of the errors named above:

```
 curl -H "@auth" -iX RENAME http://127.0.0.1:8080/Jobs/Job_new/Job2.
HTTP/1.1 406 Not Acceptable

[
{"ERROR": "The new name must not end with a period."}
]
```

SAVEAS /Jobs -> Save a copy of a job

The method `SAVEAS` allows to create a copy of a job. Therefore, the name and the new name of the job need to be stated:

```
 curl -H "@auth" -iX SAVEAS http://127.0.0.1:8080/DataSources/PG2/PG3
HTTP/1.1 200 OK

{"Success":"Job 'MyJob3' saved as 'MyJob4'"}
```

The possible error messages are the same as for `RENAME`.

START /Jobs -> Activate jobs

```
 curl -H "@auth" -iX START http://127.0.0.1:8080/Jobs/Job1
HTTP/1.1 200 OK

[
{"SUCCESS": "The job 'Job1' has been successfully activated."}
]
```

To display active jobs, the context `/Processes` is to be used, because activating a job means creating an active process which has set `ProcessType = Job`.

```
 curl -H "@auth" -iX GET http://127.0.0.1:8080/Processes/active
HTTP/1.1 200 OK
```

```
{"ProcessName": "Job1",
"AgentName": "Agent_Linux",
"ProcessType": "Job",
"StartTime": "2022-11-11-14.06.35.207742",
"ActiveFor": "2 minutes, 25 seconds",
"ProcessState": "RUNNING",
"ReturnCode": "",
"ErrorCode": 0}
```

To deactivate a job, `STOP /Processes/active/<Name>/<PID>` needs to be called.

404 Not Found - no job with stated name

```
 curl -H "@auth" -iX START http://127.0.0.1:8080/Jobs/Job2
HTTP/1.1 404 Not Found

[
{"ERROR": "Stated job 'Job2' not found"}
]
```

DELETE /Jobs -> Delete jobs

`DELETE` is for deletion of a job:

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/Jobs/Job1
HTTP/1.1 200 OK

[
{"SUCCESS": "Job 'Job1' deleted"}
]
```

404 Not Found - no job with stated name

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/Jobs/Job2
HTTP/1.1 404 Not Found

[
{"ERROR": "Stated job 'Job2' not found"}
]
```

500 Internal Server Error - server side error

If a deletion call was formally correct, but failed due to server side reasons, this output results, including a specific error message:

```
 curl -H "@auth" -iX DELETE http://127.0.0.1:8080/Jobs/Job1
```

```
HTTP/1.1 500 Internal Server Error

[
{"ERROR": "<Error message>",
"SQL ErrorCode": <Error code>}
]
```

GET Jobs/events -> Status information for Events

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/Jobs/events
HTTP/1.1 200 OK

{"Results": [{
"JobName":"MyJob1",
"EventId":"EVT00029",
"GroupId":"DEFAULT",
"AgentName":"Agent-Linux",
"FirstStart":"2024-05-06-13.50.05.684328",
"NextStart":"0000-00-00-00.00.00.000000",
"LastStart":"2024-05-06-13.50.05.684329",
"LastEnd":"2024-05-06-13.53.47.507627",
"LastRC":"00000000000",
"Restart":" ",
"StarterAgent":"",
"StarterName":"",
"StarterPID":""},
...
]}
```

By stating a specific Job, only its Events are output. If an Event is stated additionally, only this Event is printed:

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/Jobs/events/Job1/EVT00029
```

If no Event exists:

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/Jobs/events
HTTP/1.1 200 OK

{"Information": "No Events found for any Job."}
```

STOP Jobs/events -> Deactivate Job

A Job that was activated via START Jobs/defined can be deactivated this way:

```
curl -H "@auth" -iX STOP http://127.0.0.1:8080/Jobs/events/MyJob1/EVT00048
HTTP/1.1 200 OK

{"Success": "The Event 'EVT00048' has been successfully unscheduled."}
```

404 Not Found – No such Event

If the Event was not found:

```
curl -H "@auth" -iX STOP http://127.0.0.1:8080/Jobs/events/MyJob1/EVT0004
HTTP/1.1 404 NOT Found

{"Error": "Stated Event 'EVT0004' for Job 'MyJob1' not found."}
```

412 Precondition Failed – No such Job

If the Job was not found:

```
curl -H "@auth" -iX STOP http://127.0.0.1:8080/Jobs/events/MyJob9/EVT00048
412 Precondition Failed

{"Error": "Stated Job 'MyJob9' not found."}
```

Resource InputObjects (Repository input objects)

Using the resource InputObjects repository definitions of input tables can be retrieved. This includes the definitions of the tables themselves, as well as the definitions that are stored in any additional repository objects linked to the table (SQL history, field tables, ...).

OPTIONS InputObjects -> Method Overview

```
curl -H "@auth" -iX OPTIONS http://127.0.0.1:8080/InputObjects
HTTP/1.1 200 OK

{"= HTTP method =              ":"= Explanation =",
"GET    /InputObjects         ":"Get information for input table(s).",
"POST   /InputObjects         ":"Add a new or replace an input object",
"PUT    /InputObjects         ":"Change an input object",
"DELETE /InputObjects         ":"Delete an input object",
"= Optional queries for GET = ":"= Explanation =",
"groupid=<GroupId>              ":"Return tables for group <GroupId> only (default value = 'DEFA
ULT')",
"sourcetype=<SourceType>      ":"Return tables for type <SourceType> only",
"sourcename=<SourceName>      ":"Return tables with name <SourceName> only",
"validfrom=<ValidFrom>        ":"Return tables valid at <ValidFrom> only",
"details=<Y|N>                ":"Returned result: 'N' key fields only, 'Y' all fields (default v
alue = 'N')",
"= Request data =             ":"= Explanation =",
"Data in JSON format          ":"Use data to specify the attributes of the object(s) to insert/u
pdate/delete"}
```

GET InputObjects -> Information about input tables

The simple call without further query parameters returns all input tables of the 'DEFAULT' (standard) group as a result.

The result is returned in JSON notation. It informs whether the output is with all details or not and the individual input tables as JSON objects within a JSON array named ‘Results’. The key names of the key-value pairs in these objects correspond to the names of the fields of the repository objects (here, for example, the field names '**GroupId**', '**SourceName**' and '**ValidFrom**' of the repository table '**InputTables**').

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/InputObjects
HTTP/1.1 200 OK

{"Details": false,
"Results": [
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00001.00001",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"
},
...
{"GroupId": "DEFAULT",
"SourceType": "VSAM",
"SourceName": "VSAM.KVBEST.FILE",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

The call can be restricted by specifying additional parameters in the query data. Only the tables that match the criteria of the parameters are returned.

The following parameters are available:

Parameter	Description
GroupId	Restricts the determined tables to a group (the default is 'DEFAULT')
sourcetype	Restricts the determined tables to a source type
sourcename	Restricts the determined tables to a table name
validfrom	Restricts the determined tables to a timestamp at which the table must be valid
details	Used to set the level of detail of the return. The possible values are: Y: all fields of the repository object N: only the key fields of the repository object (default)

Note:

- If all restrictive parameters ('groupid', 'sourcetype', 'sourcename' and 'validfrom') are set, 'Y' is automatically used for 'details'. In this case the result only consists of one input table with all the information from the repository definition of this table.
- The meaning of “all the information from the repository definition” is the information that is relevant to the corresponding table. For example, for an Oracle table, all type-specific information that is not related to Oracle (e.g. the repository fields 'DatacomElements', 'Db2TableName', 'VSAMKeyLen', ...) will be suppressed.
- If these parameters contain special characters such as spaces, <, >, ^, $, etc., they must be encoded accordingly before sending to the REST API server. A table with some characters and their encoding can be found in the Resource Import chapter.

Examples of using the parameters

Only the Adabas tables should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/InputObjects?sourcetype=ADABAS
HTTP/1.1 200 OK

{"Details": false,
"Results": [
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00001.00001",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"
},
...
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "01111.01111",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

Only the Oracle table “ORCL.DEMO.ARTIKEL” should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/InputObjects?sourcetype=ORACLE&sourcename=ORCL.DEM
O.ARTIKEL
HTTP/1.1 200 OK

{"Details": false,
"Results": [
{"GroupId": "DEFAULT",
"SourceType": "ORACLE",
"SourceName": "ORCL.DEMO.ARTIKEL",
"ValidFrom": "2020-07-15-08.29.21.000000",
"ValidUntil": "2021-07-15-08.29.20.999999"
},
{"GroupId": "DEFAULT",
"SourceType": "ORACLE",
"SourceName": "ORCL.DEMO.ARTIKEL",
"ValidFrom": "2021-07-15-08.29.21.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

Note: The table was reimported, which is why two versions of the table exist with different validity ranges.

Only the Oracle table “ORCL.DEMO.ARTIKEL”, valid at time “2021-07-15-08.29.21.000000”, should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/InputObjects?sourcetype=ORACLE&sourcename=ORCL.DEM
O.ARTIKEL&
validfrom=2021-07-15-08.29.21.000000
HTTP/1.1 200 OK

{"Details": true,
"Results": [
{"GroupId": "DEFAULT",
"SourceType": "ORACLE",
"SourceName": "ORCL.DEMO.ARTIKEL",
"ValidFrom": "2021-07-15-08.29.21.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
"AgentName": "",
"AliasSourceType": "",
"AliasSourceName": "",
"ProjectName": "",
"ImporterVersion": "0101001",
"DefaultCCSID": 1252,
"DblDefaultCCSID": 1208,
"UniqueKeyFields": "NR",
"RecordExitFields": "",
"RecordExitCode": "",
"BulkRecordCount": 0,
"BRCDate": "",
"InternalUse": "",
"SQL Histories": [
{"GroupId": "DEFAULT",
"SourceType": "ORACLE",
"SourceName": "ORCL.4711",
"ValidFrom": "2020-07-15-08.29.21.000000",
"ValidUntil": "2021-07-15-08.29.20.999999",
"Description": "",
"DatabaseName": "ORCL",
"TCreator": "DEMO",
"TName": "ARTIKEL",
"TableTimestamp": "2020-07-15-08.29.21.000000",
"LFD_DbSpaceNo": 0,
"LFD_Tabid": 0
},
{"GroupId": "DEFAULT",
"SourceType": "ORACLE",
"SourceName": "ORCL.471215",
"ValidFrom": "2021-07-15-08.29.21.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
"DatabaseName": "ORCL",
"TCreator": "DEMO",
"TName": "ARTIKEL",
"TableTimestamp": "2021-07-15-08.29.21.000000",
"LFD_DbSpaceNo": 0,
"LFD_Tabid": 0}]}]}
```

Note: This example is for an Oracle table. A SQL history is maintained in the repository for this source type. Therefore, and because the parameter 'details=Y' is automatically set by defining all other parameters, the output is supplemented with the associated entries from the repository table 'InputTablesSQLHist' (see chapter Source type-dependent additional information).

Source type-dependent additional information

In addition to the input tables, more tables with additional information are available in the repository for some source types. This section describes these source types, the corresponding additional repository tables, and how they appear in the result output.

This additional information is only displayed in the result if the parameter 'details=Y' is set, or if it is automatically activated by setting all other parameters.

Source type ‚ADABAS‘

For this source type, the information from the repository table “InputTablesAdaLF” is also evaluated and included in the result as additional JSON objects within an JSON array called “Adabas LF”.

```
{"Details": true,
"Results": [
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00001.00001",
...
"AdabasIsnOffset": 0,
"AdabasLOBFileNumber": 0,
"Adabas LF": [
{"GroupId": "DEFAULT",
"SourceName": "00001.00001",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
"AdabasLF": "F,1,AA, 8,A,129,000;F,1,AB, 0, ,000,000;F,2,AC,20,A,016,000;"}]
...
```

Source type ‚DLI‘

For this source type, the information from the repository table “InputTablesDLIAKey” is also evaluated and included in the result as additional JSON objects within a JSON array called “DLI AKeys”.

```
{"Details": false,
"Results": [
{"GroupId": "DEFAULT",
"SourceType": "DLI",
"SourceName": "DBD03.S0314",
...
"DLISegmentVariable": "N",
"DLISSA": "S0300   (APLID   >=:key:8),S0314   (ZEIN    >=:key:4).",
"DLI AKeys": [
{"GroupId": "DEFAULT",
"SourceName": "DBD03.S0314",
"AltKeyPSB": "PS3A203",
"AltKeyPCB": 1,
"ValidFrom": "2017-06-28-10.23.25.627206",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
"AltKeyUnique": "Y",
"AltKeySegKeyLen": 4,
"AltKeyConcatLen": 20,
"AltKeyFields": "KFA_ZEIN(12;4),KFA_S0300(16;8)"},
...
{"GroupId": "DEFAULT",
"SourceName": "DBD03.S0314",
"AltKeyPSB": "PS3A203",
"AltKeyPCB": 5,
"ValidFrom": "2017-06-28-10.23.25.627206",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
"AltKeyUnique": "Y",
"AltKeySegKeyLen": 57,
"AltKeyConcatLen": 65,
"AltKeyFields": "MATID,BAUM,SACHNR,APLID,KFA_S0300(57;8)"}]}]}
```

Source type ‚IDMS‘

For this source type, the information from the repository tables “InputTablesIDMSOD” and “InputTablesIDMSPD” is also evaluated and included in the result as additional JSON objects within JSON arrays called “IDMS Owner Data” respectively “IDMS Path Data”.

```
{"Details": true,
"Results": [
{"GroupId": "DEFAULT",
"SourceType": "IDMS",
"SourceName": "AINAB02.RINA-SPECIALS-DB",
...
"IDMSControlLen": -1,
"IDMS Owner Data": [
{"GroupId": "DEFAULT",
"SourceName": "AINAB02.RINA-SPECIALS-DB",
"OwnerPathSeq": 1,
...
"NextKeySQL": ""
},
{"GroupId": "DEFAULT",
"SourceName": "AINAB02.RINA-SPECIALS-DB",
"OwnerPathSeq": 2,
...
"NextKeySQL": ""}
],
"IDMS Path Data": [
{"GroupId": "DEFAULT",
"SourceName": "AINAB02.RINA-SPECIALS-DB",
"OwnerPathSeq": 1,
...
"UpdateQuerySQL": ""
},
{"PD_2": [
{"GroupId": "DEFAULT",
"SourceName": "AINAB02.RINA-SPECIALS-DB",
"OwnerPathSeq": 2,
...
"UpdateQuerySQL": ""}]}]}
```

Source types ‚DB2‘, ‚ORACLE‘, ‚MSSQL‘, ‚INFORMIX‘, ‚POSTGRES‘, ‚MYSQL‘ and ‚MARIADB‘

For these source types, the information from the repository table “InputTablesSQLHist” is also evaluated and included in the result as additional JSON objects within a JSON array called “SQL Histories”.

```
{"Details": true,
"Results": [
{"GroupId": "DEFAULT",
"SourceType": "ORACLE",
"SourceName": "ORCL.DEMO.ARTIKEL",
...
"InternalUse": "",
"SQL Histories": [
{"GroupId": "DEFAULT",
"SourceType": "ORACLE",
"SourceName": "ORCL.4711",
...
"LFD_Tabid": 0
},
{"GroupId": "DEFAULT",
"SourceType": "ORACLE",
"SourceName": "ORCL.471215",
...
"LFD_Tabid": 0}]}]}
```

Various source types with support for repeating fields

These source types are, for example, 'ADABAS' using 'Periodic Groups' or 'Multiple Fields', or 'VSAM' and 'FILE' (bound to a Cobol copybook using OCCURS fields). In the repository maintenance of the dashboard, such input tables can be identified by green fields in the field list.

For tables of this type of source, the information from the repository table “InputFieldTables” is also evaluated and included in the result as additional JSON objects within a JSON array called “Input Field Tables”.

```
HTTP/1.1 200 OK

{"Details": true,
"Results": [
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00001.00001",
...
"AdabasLOBFileNumber": 0,
"Adabas LF": [
{"GroupId": "DEFAULT",
"SourceName": "00001.00001",
...
"AdabasLF": "F,1,AA, 8,A,129,000;F,1,AB, 0, ,000,000;F,2,AC,20,A,016,000;"}
],
"Input Field Tables": [
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00001.00001",
"TableSequence": 1,
"TableName": "ADABAS_AI",
...
"MaxOccurs": 3,
"UniqueKey": ""},
...
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00001.00001",
"TableSequence": 5,
"TableName": "ADABAS_AZ",
...
"MaxOccurs": 4,
"UniqueKey": ""}]}]}
```

POST InputObjects -> Inserting input tables

Calling this method allows to insert an input table and other objects, related to input tables (`InputTablesAdaLF`, `InputTablesDLIAKey`, `InputTablesIDMSOD`, `InputTablesIDMSPD`, `InputTablesSQLHist` and `InputFieldTables`), into the repository.

For the insert all field values of the objects must be stored in a file in JSON format, and this file is then sent in the request data. The names of the field values correspond to the field names in the corresponding repository tables, and are not case sensitive.

The individual objects are grouped into JSON arrays. The identifiers “InputTables”, “Adabas LF”, “Input Field Tables”, “DLI Akeys”, “IDMS Owner Data”, “IDMS Path Data” and “SQL Histories” are available for these groups, and are also not case sensitive.

Two tables are to be inserted into the repository table InputTables. To do this, a JSON file “data.json” with the following content is created.

```
{"InputTables": [
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00001.00001",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
...
"AdabasIsnOffset": 0,
"AdabasLOBFileNumber": 0
},
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00002.00002",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
...
"AdabasIsnOffset": 0,
"AdabasLOBFileNumber": 0
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/InputObjects"
HTTP/1.1 200 OK
```

```
{"Input tables inserted":[
{"GroupId":"DEFAULT","SourceType":"ADABAS","SourceName":"00001.00001","ValidFrom":"0000-00-00-0
0.00.00.000000","ValidUntil":"9999-99-99-99.99.99.999999"},
{"GroupId":"DEFAULT","SourceType":"ADABAS","SourceName":"00002.00002","ValidFrom":"0000-00-00-0
0.00.00.000000","ValidUntil":"9999-99-99-99.99.99.999999"}
]}
```

A table is to be inserted into the repository table InputTables. At the same time, the related tables `InputTablesAdaLF` and `InputFieldTables` should receive an entry. To do this, a JSON file “data.json” with the following content (shown in abbreviated form) is created.

```
{
"InputTables": [
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00001.00001",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
...
"AdabasIsnOffset": 0,
"AdabasLOBFileNumber": 0
} ],
"Adabas LF": [
{"GroupId": "DEFAULT",
"SourceName": "00001.00001",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
"AdabasLF": "F,1,AA, 8,A,129,000;F,1,AB, 0, ,000,000;F,2,AC,20,A,016,000;F,2,AE,20,A,134,00
0;F,2,AD,20,A,016,000;F,1,AF, 1,A,064,000;F,1,AG, 1,A,064,000;F,1,AH, 6,P,128,001;F,1,A1, 0, ,00
0,000;F,2,AI,20,A,048,000;"
} ],
"Input Field Tables": [
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00001.00001",
"TableSequence": 1,
"TableName": "ADABAS_AI",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
"MaxOccurs": 3,
"ActOccursField": "ADABAS_AI_C",
"TabIndexField": "ADABAS_AI_TAB_IND",
"SubTableOf": "00001.00001",
"UniqueKey": ""
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/InputObjects"
HTTP/1.1 200 OK
```

```
{"Input tables inserted":[
{"GroupId":"DEFAULT","SourceType":"ADABAS","SourceName":"00001.00001","ValidFrom":"0000-00-00-0
0.00.00.000000","ValidUntil":"9999-99-99-99.99.99.999999"}
],"Input field tables inserted":[
{"GroupId":"DEFAULT","SourceType":"ADABAS","SourceName":"00001.00001","TableSequence":1,"TableNa
me":"ADABAS_AI","ValidFrom":"0000-00-00-00.00.00.000000","ValidUntil":"9999-99-99-99.99.99.99999
9"}
],"Adabas LF inserted":[
{"GroupId":"DEFAULT","SourceName":"00001.00001","ValidFrom":"0000-00-00-00.00.00.000000","ValidU
ntil":"9999-99-99-99.99.99.999999"}
]}
```

PUT InputObjects -> Updating input tables

Calling this method allows to update an input table and other objects, related to input tables (`InputTablesAdaLF`, `InputTablesDLIAKey`, `InputTablesIDMSOD`, `InputTablesIDMSPD`, `InputTablesSQLHist` and `InputFieldTables`), in the repository.

To update, the key values of the object to be updated as well as all field values of the resulting object must be saved in a file in JSON format, and this file is then sent in the request data. The names of the field values correspond to the field names in the corresponding repository tables, and are not case sensitive. The names of the key values of the object to be updated is preceded by the prefix “Before_”.

The individual objects are grouped into JSON arrays. The identifiers “InputTables”, “Adabas LF”, “Input Field Tables”, “DLI Akeys”, “IDMS Owner Data”, “IDMS Path Data” and “SQL Histories” are available for these groups, and are also not case sensitive.

Note: When key values of an input table are updated, all repository objects related to the input table through these key values (`InputTablesAdaLF`, `InputTablesDLIAKey`, `InputTablesIDMSOD`, `InputTablesIDMSPD`, `InputTablesSQLHist`, `InputFieldTables`, `InputFields`, `ReplicationRules` and `ReplicationLinkages`) are also automatically updated so that the relationship of these objects to one another is not lost. It is therefore not necessary to additionally update these dependent objects.

A table in the repository table InputTables is to be updated. To do this, a JSON file “data.json” with the following content (shown in abbreviated form) is created.

```
{
"InputTables": [
{"Before_GroupId": "DEFAULT",
"Before_SourceType": "ADABAS",
"Before_SourceName": "00001.00001",
"Before_ValidFrom": "0000-00-00-00.00.00.000000",
"Before_ValidUntil": "9999-99-99-99.99.99.999999",
"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00002.00002",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "Renamed table",
...
"AdabasIsnOffset": 1,
"AdabasLOBFileNumber": 2,
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/InputObjects"
HTTP/1.1 200 OK
{"Input tables updated":[
{"GroupId":"DEFAULT","SourceType":"ADABAS","SourceName":"00001.00001","ValidFrom":"0000-00-00-0
0.00.00.000000","ValidUntil":"9999-99-99-99.99.99.999999"}
]}
```

A table in the repository table InputTables is to be updated. At the same time, the related tables InputTablesAdaLF and InputFieldTables should be updated. To do this, a JSON file “data.json” with the following content (shown in abbreviated form) is created.

```
{"InputTables": [
{"Before_GroupId": "DEFAULT",
"Before_SourceType": "ADABAS",
"Before_SourceName": "00001.00001",
"Before_ValidFrom": "0000-00-00-00.00.00.000000",
"Before_ValidUntil": "9999-99-99-99.99.99.999999",
"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00001.00001",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "Changing LOB File Number",
...
"AdabasIsnOffset": 1,
"AdabasLOBFileNumber": 42,
} ],
"Adabas LF": [
{"Before_GroupId": "DEFAULT",
"Before_SourceName": "00001.00001",
"Before_ValidFrom": "0000-00-00-00.00.00.000000",
"Before_ValidUntil": "9999-99-99-99.99.99.999999",
"GroupId": "DEFAULT",
"SourceName": "00001.00001",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "Added field AJ",
"AdabasLF": "F,1,AA, 8,A,129,000;F,1,AB, 0, ,000,000;F,2,AC,20,A,016,000;F,2,AE,20,A,134,00
0;F,2,AD,20,A,016,000;F,1,AF, 1,A,064,000;F,1,AG, 1,A,064,000;F,1,AH, 6,P,128,001;F,1,A1, 0, ,00
0,000;F,2,AI,20,A,048,000;F,2,AJ,20,A,048,000;"
} ],
"Input Field Tables": [
{"Before_GroupId": "DEFAULT",
"Before_SourceType": "ADABAS",
"Before_SourceName": "00001.00001",
"Before_TableSequence": 1,
"Before_TableName": "ADABAS_AI",
"Before_ValidFrom": "0000-00-00-00.00.00.000000",
"Before_ValidUntil": "9999-99-99-99.99.99.999999",
"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00001.00001",
"TableSequence": 1,
"TableName": "ADABAS_AI",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "Changed Max Occurs to 6",
"MaxOccurs": 6,
"ActOccursField": "ADABAS_AI_C",
"TabIndexField": "ADABAS_AI_TAB_IND",
"SubTableOf": "00001.00001",
"UniqueKey": ""
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/InputObjects"
HTTP/1.1 200 OK

{"Input tables updated":[
{"GroupId":"DEFAULT","SourceType":"ADABAS","SourceName":"00001.00001","ValidFrom":"0000-00-00-0
0.00.00.000000","ValidUntil":"9999-99-99-99.99.99.999999"}
],"Input field tables updated":[
{"GroupId":"DEFAULT","SourceType":"ADABAS","SourceName":"00001.00001","TableSequence":1,"TableNa
me":"ADABAS_AI","ValidFrom":"0000-00-00-00.00.00.000000","ValidUntil":"9999-99-99-99.99.99.99999
9"}
],"Adabas LF updated":[
{"GroupId":"DEFAULT","SourceName":"00001.00001","ValidFrom":"0000-00-00-00.00.00.000000","ValidU
ntil":"9999-99-99-99.99.99.999999"}
]}
```

DELETE InputObjects -> Deleting input tables

Calling this method allows to delete an input table and other objects, related to input tables (`InputTablesAdaLF`, `InputTablesDLIAKey`, `InputTablesIDMSOD`, `InputTablesIDMSPD`, `InputTablesSQLHist` and `InputFieldTables`), from the repository.

To delete, the key values of the object to be deleted must be saved to a file in JSON format, and this file is then sent in the request data. The names of the key values correspond to the field names in the corresponding repository tables, and are not case sensitive.

The individual objects are grouped into JSON arrays. The identifiers “InputTables”, “Adabas LF”, “Input Field Tables”, “DLI Akeys”, “IDMS Owner Data”, “IDMS Path Data” and “SQL Histories” are available for these groups, and are also not case sensitive.

Note: When an input table is deleted, all repository objects related to the input table (`InputTablesAdaLF`, `InputTablesDLIAKey`, `InputTablesIDMSOD`, `InputTablesIDMSPD`, `InputTablesSQLHist`, `InputFieldTables`, `InputFields`, `ReplicationRules` and `ReplicationLinkages`) are also automatically deleted, since the existence of these objects makes no sense without the input table. It is therefore not necessary to additionally delete these dependent objects.

A table is to be deleted from the repository table InputTables. To do this, a JSON file “data.json” with the following content is created.

```
{
"InputTables": [
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00001.00001",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX DELETE "http://127.0.0.1:8080/InputObjects"
HTTP/1.1 200 OK

{"Input tables deleted":[
{"GroupId":"DEFAULT","SourceType":"ADABAS","SourceName":"00001.00001","ValidFrom":"0000-00-00-0
0.00.00.000000","ValidUntil":"9999-99-99-99.99.99.999999"}
]}
```

Entries should be deleted from the `InputTablesAdaLF` and `InputFieldTables` tables. To do this, a JSON file “data.json” with the following content is created.

```
{
"Adabas LF": [
{"GroupId": "DEFAULT",
"SourceName": "00001.00001",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"
} ],
"Input Field Tables": [
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00001.00001",
"TableSequence": 1,
"TableName": "ADABAS_AI",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX DELETE "http://127.0.0.1:8080/InputObjects"
HTTP/1.1 200 OK

{"Input field tables deleted":[
{"GroupId":"DEFAULT","SourceType":"ADABAS","SourceName":"00001.00001","TableSequence":1,"TableNa
me":"ADABAS_AI","ValidFrom":"0000-00-00-00.00.00.000000","ValidUntil":"9999-99-99-99.99.99.99999
9"}
],"Adabas LF deleted":[
{"GroupId":"DEFAULT","SourceName":"00001.00001","ValidFrom":"0000-00-00-00.00.00.000000","ValidU
ntil":"9999-99-99-99.99.99.999999"}
]}
```

Error messages for resource InputObjects

If an error occurs when executing an option of resource InputObjects, an HTTP return code not equal to 200 is returned.

Below is a brief overview of possible error messages and their causes:

```
curl -H "@auth" -iX GETT http://127.0.0.1:8080/InputObjects
HTTP/1.1 400 Bad Request

{"Error": "Method 'GETT' is not supported"}
```

An incorrect method was used in the call. The call needs to be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/InputObjects?groupid=MyGroup
HTTP/1.1 400 Bad Request

{"Error": "InputObjects failed",
"Reason": "Specified groupid 'MyGroup' is not defined in repository."}
```

A group was used in the call that is not defined in the repository. The group to be used must be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/InputObjects?
validfrom=24-03-01-12.34.56.123456
HTTP/1.1 400 Bad Request

{"Error": "InputObjects failed",
"Reason": "Specified timestamp '24-03-01-12.34.56.123456' has invalid format. Valid format is 'Y
YYY-MM-DD-HH.MM.SS.FFFFFF'"}
```

A timestamp with an invalid format was used in the call. The timestamp to be used must be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/InputObjects ...
HTTP/1.1 500 Internal Server Error

{"Error": "InputObjects failed",
"Reason": "Caught exception while creating result",
"Exception": "<Exception message>"}
```

If there is a problem with the internal processing of a call, this message will be returned. The placeholder <Exception message> is replaced by a message that corresponds to the problem.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/InputObjects"
HTTP/1.1 500 Internal Server Error

{"Error":"InputObjects failed",
"Reason":"Got SQL error message while inserting ",
"Object":"InputTable DEFAULT.ADABAS.00001.00001 (0000-00-00-00.00.00.000000 – 9999-99-99-99.99.9
9.999999)",
"SQL message":"TCS0597E;70R0,WIN64_6.2.9200; Repository SQL error: TCS0256E;70R0,WIN64_6.2.9200;
PostgreSQL libpq SQLSTATE 23505, ERROR: duplicate key value violates unique constraint »inputtab
les_pkey« DETAIL: Key »(groupid, sourcetype, sourcename, validfrom, validuntil)=(DEFAULT , ADABA
S , 00001.00001 , 0000-00-00-00.00.00.000000, 9999-99-99-99.99.99.999999)« already exists.; N/
A."}
```

If there are problems (SQL errors) when inserting, updating or deleting objects in the repository, this message is returned. The affected object is given as well as the error message received from the database.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/InputObjects"
HTTP/1.1 400 Bad Request

{"Error":"InputObjects failed",
"Reason":"Updating 'InputTable' object in repository not possible, because not all key values of
'Before' object have been specified.",
"Object":"InputTable DEFAULT.ADABAS.00001.00001 ( 0000-00-00-00.00.00.000000 - )"}
```

If not all key values are provided to uniquely identify an object, this will be indicated with a message like this.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/InputObjects"
HTTP/1.1 400 Bad Request

{"Error":"InputObjects failed",
"Reason":"Updating 'InputTable' object in repository not possible, because object does not exist
in repository.",
"Object":" InputTable DEFAULT.ADABAS.00001.00001 ( 0000-00-00-00.00.00.000000 - 9999-99-99-99.9
9.99.999999)"}
```

If no object matching the key values passed is found in the repository, this will be indicated with a message like this.

Resource OutputObjects (Repository output objects)

Using the resource OutputObjects repository definitions of output tables can be retrieved. This includes the definitions of the tables themselves, as well as the definitions that are stored in any additional repository objects linked to the table (field tables and IDMS paths).

OPTIONS OutputObjects -> Method Overview

```
curl -H "@auth" -iX OPTIONS http://127.0.0.1:8080/OutputObjects
HTTP/1.1 200 OK

{"= HTTP method =               ":"= Explanation =",
"GET    /OutputObjects         ":"Get information for output table(s)",
"POST   /OutputObjects         ":"Add a new or delete an output object",
"PUT    /OutputObjects         ":"Change an output object",
"DELETE /OutputObjects         ":"Delete an output object",
"= Optional queries for GET =  ":"= Explanation =",
"groupid=<GroupId>               ":"Return tables for group <GroupId> only (default value = 'DEF
AULT')",
"outputtargetname=<TargetName> ":"Return tables for target <TargetName> only",
"outputtable=<OutputTable>     ":"Return tables with name <OutputTable> only",
"validfrom=<ValidFrom>         ":"Return tables valid at <ValidFrom> only",
"details=<Y|N>                 ":"Returned result: 'N' key fields only,
 'Y' all fields (default value = 'N')",
"= Request data =              ":"= Explanation =",
"Data in JSON format           ":"Use data to specify the attributes of the object(s) to insert/
update/delete"}
```

GET OutputObjects -> Information about output tables

The simple call without further query parameters returns all output tables of the 'DEFAULT' (standard) group as a result.

The result is returned in JSON notation. It informs whether the output is with all details or not and the individual output tables as JSON objects within a JSON array named 'Results'. The key names of the key-value pairs in these objects correspond to the names of the fields of the repository objects (here, for example, the field names 'GroupId', ‚OutputTable‘ and ‚ValidFrom‘ of the repository table 'OutputTables').

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputObjects
HTTP/1.1 200 OK

{"Details": false,
"Results": [
{"GroupId": "DEFAULT",
"OutputTargetName": "Snowflake_DB",
"OutputTable": "PUBLIC.ZWOELF_ALLTYPES",
"ValidFrom": "2022-04-22-13.15.20.500000",
"ValidUntil": "9999-99-99-99.99.99.999999"}],
...
{"GroupId": "DEFAULT",
"OutputTargetName": "Target_File",
"OutputTable": "demo.ddltest",
"ValidFrom": "2023-07-17-12.19.22.500000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

The call can be restricted by specifying additional parameters in the query data. Only the tables that match the criteria of the parameters are returned.

The following parameters are available:

Parameter	Description
groupid	Restricts the determined tables to a group (the default is 'DEFAULT')
outputtargetname	Restricts the determined tables to an Output Target
outputtable	Restricts the determined tables to a table name
validfrom	Restricts the determined tables to a timestamp at which the table must be valid
details	Used to set the level of detail of the return. The possible values are: Y: all fields of the repository object N: only the key fields of the repository object (default)

Note:

- If all restrictive parameters ('groupid', 'outputtargetname‘, ‚outputtable‘ and 'validfrom') are set, 'Y' is automatically used for 'details'. In this case the result only consists of one output table with all the information from the repository definition of this table.
- The meaning of “all the information from the repository definition” is the information that is relevant to the corresponding table. For example, for an Adabas table, all type-specific information that is not related to Adabas (e.g. the repository fields ‚DLIDatabaseName‘, ‚DatacomElements‘, ‚IDMSControlField‘, …) will be suppressed.
- If these parameters contain special characters such as spaces, <, >, ^, $, etc., they must be encoded accordingly before sending to the REST API server. A table with some characters and their encoding can be found in chapter „Resource Import“.

Examples of using the parameters

Only the tables of the Output Target “Target_Postgres” should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputObjects?outputtargetname=Target_Postgres
HTTP/1.1 200 OK

{"Details": false,
"Results": [
{"GroupId": "DEFAULT",
"OutputTargetName": "Target_Postgres",
"OutputTable": "public.account",
"ValidFrom": "2021-01-18-13.39.23.500000",
"ValidUntil": "9999-99-99-99.99.99.999999"},
...
{"GroupId": "DEFAULT",
"OutputTargetName": "Target_Postgres",
"OutputTable": "public.zahlung",
"ValidFrom": "2021-07-09-07.14.57.727000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

Only the table “public.artikel” of Output Target “Target_Postgres” should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputObjects?outputtargetname=Target_Postgres&out
puttable=public.artikel
HTTP/1.1 200 OK

{"Details": false,
"Results": [
{"GroupId": "DEFAULT",
"OutputTargetName": "Target_Postgres",
"OutputTable": "public.artikel",
"ValidFrom": "2021-12-14-13.41.36.500000",
"ValidUntil": "2023-10-31-09.34.37.499999"
},
{"GroupId": "DEFAULT",
"OutputTargetName": "Target_Postgres",
"OutputTable": "public.artikel",
"ValidFrom": "2023-10-31-09.34.37.500000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

Note: The table was reimported, which is why two versions of the table exist with different validity ranges.

Only the table “public.artikel” of Output Target “Target_Postgres”, valid at time “2021-07-15-08.29.21.000000”, should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputObjects?outputtargetname=Target_Postgres&out
puttable=public.artikel&
validfrom=2023-10-31-09.34.37.500000
HTTP/1.1 200 OK

{"Details": true,
"Results": [
{"GroupId": "DEFAULT",
"OutputTargetName": "Target_Postgres",
"OutputTable": "public.artikel",
"ValidFrom": "2023-10-31-09.34.37.500000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
"UniqueKeyFields": "",
"SupEmptyRowIns": "N",
"KeyConstraintTo": "",
"DeleteCascadeInDB": "N",
"ClearDataPreBulk": "Y",
"IndexDrpPreBulk": "N",
"IndexDrpDDLPreBulk": "",
"CommitAfterDropDDL": "N",
"IndexCrtAfterBulk": "N",
"IndexCrtDDLAftBulk": "",
"CommitPreCrtDDL": "N",
"UserKeyFields01": "",
"UserKeyFields02": "",
"UserKeyFields03": "",
"UserKeyFields04": "",
"UserKeyFields05": "",
"InTableSpace": "",
"DefaultCCSID": 1252,
"DblDefaultCCSID": 0,
"LastChange": "2024-02-02-13.56.05.483000"}]}
```

Note: By defining all other parameters, the parameter 'details=Y' is automatically set and the output contains all relevant information about the table.

Output Target type-dependent additional information

In addition to the output tables, more tables with additional information are available in the repository for some Output Target types. This section describes these target types, the corresponding additional repository tables, and how they appear in the result output.

This additional information is only displayed in the result if the parameter 'details=Y' is set, or if it is automatically activated by setting all other parameters.

Target type ‚IDMS'

For this target type, the information from the repository table "OutputIDMSPaths" is also evaluated and included in the result as additional JSON objects within a JSON array called "IDMS Path Data".

```
HTTP/1.1 200 OK

{"Details": true,
"Results": [
{"GroupId": "DEFAULT",
"OutputTargetName": "IDMS_TARGET",
"OutputTable": "DENTAL_CLAIMS",
...
"IDMSAreas": "INS-DEMO-REGION",
"IDMS Path Data": [
{"GroupId": "DEFAULT",
"OutputTargetName": "IDMS_TARGET",
"OutputTableName": "DENTAL_CLAIMS",
"PathNumber": 1,
"PathSequence": 1,
...
"SetToNextMember": "EMP-COVERAGE",
"FieldsToNextMember": ""
},
{"GroupId": "DEFAULT",
"OutputTargetName": "IDMS_TARGET",
"OutputTableName": "DENTAL_CLAIMS",
"PathNumber": 1,
"PathSequence": 2,
...
"SetToNextMember": "COVERAGE-CLAIMS",
"FieldsToNextMember": ""}]}]}
```

Various target types with support for repeating fields

These types are, for example, 'ADABAS' with 'Periodic groups' or 'Multiple Fields', or 'POSTGRESQL' (with array type fields). In the repository management of the dashboard, such output tables can be identified by green fields in the field list.

For tables of this type of target, the information from the repository table “OutputFieldTables” is also evaluated and included in the result as additional JSON objects within a JSON array called “Output Field Tables”.

```
HTTP/1.1 200 OK

{"Details": true,
"Results": [
{"GroupId": "DEFAULT",
"OutputTargetName": "Adabas_zos",
"OutputTable": "T00011_00011",
...
"AdabasUserIsn": "N",
"Output Field Tables": [
{"GroupId": "DEFAULT",
"OutputTargetName": "Adabas_zos",
"OutputTableName": "T00011_00011",
"TableSequence": 1,
"TableName": "ADABAS_AI",
...
"MaxOccurs": 2,
"ActOccursField": "ADABAS_AI_C",
"SubTableOf": "T00011_00011"},
...
{"GroupId": "DEFAULT",
"OutputTargetName": "Adabas_zos",
"OutputTableName": "T00011_00011",
"TableSequence": 5,
"TableName": "ADABAS_AZ",
...
"MaxOccurs": 3,
"ActOccursField": "ADABAS_AZ_C",
"SubTableOf": "T00011_00011"}]}]}
```

POST OutputObjects -> Inserting output tables

Calling this method allows to insert an output table and other objects, related to output tables (OutputIDMSPaths and OutputFieldTables), into the repository.

For the insert all field values of the objects must be stored in a file in JSON format, and this file is then sent in the request data. The names of the field values correspond to the field names in the corresponding repository tables, and are not case sensitive.

The individual objects are grouped into JSON arrays. The identifiers “OutputTables“, “Output Field Tables“ and “IDMS Path Data“ are available for these groups, and are also not case sensitive.

Example 1
Two tables are to be inserted into the repository table OutputTables. To do this, a JSON file “data.json” with the following content (shown in abbreviated form) is created.

```
{"OutputTables":[
{"GroupId":"DEFAULT",
"OutputTargetName":"DB9D_output",
"OutputTable":"DEMO.CUSTOMERS",
"ValidFrom":"2024-04-15-10.06.59.521000",
"ValidUntil":"9999-99-99-99.99.99.999999",
"Description":"",
"UniqueKeyFields":"CUSTOMER_ID",
...
"DefaultCCSID":37,
"DblDefaultCCSID":0,
"LastChange":"2024-05-12-09.17.54.734000"
},
{"GroupId":"DEFAULT",
"OutputTargetName":"DB9D_output",
"OutputTable":"DEMO.EMPLOYEES",
"ValidFrom":"2024-02-01-11.18.36.432000",
"ValidUntil":"9999-99-99-99.99.99.999999",
"Description":"",
"UniqueKeyFields":"EMPLOYEE_ID",
...
"DefaultCCSID":37,
"DblDefaultCCSID":0,
"LastChange":"2024-04-13-08.28.30.852000"
}
]}
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/OutputObjects"
HTTP/1.1 200 OK

{"Output tables inserted":[
{"GroupId":"DEFAULT","OutputTargetName":"DB9D_output","OutputTable":"DEMO.CUSTOMERS","ValidFro
m":"2024-04-15-10.06.59.521000","ValidUntil":"9999-99-99-99.99.99.999999"},
{"GroupId":"DEFAULT","OutputTargetName":"DB9D_output","OutputTable":"DEMO.EMPLOYEES","ValidFro
m":"2024-02-01-11.18.36.432000","ValidUntil":"9999-99-99-99.99.99.999999"}
]}
```

Example 2
A table is to be inserted into the repository table OutputTables. At the same time, the related tables OutputFieldTables und OuputIDMSPaths should receive an entry. To do this, a JSON file “data.json” with the following content (shown in abbreviated form) is created.

```
{"OutputTables":[
{"GroupId":"DEFAULT",
"OutputTargetName":"DB9D_output",
"OutputTable":"DEMO.EMPLOYEES",
"ValidFrom":"2024-02-01-11.18.36.432000",
"ValidUntil":"9999-99-99-99.99.99.999999",
"Description":"",
"UniqueKeyFields":"EMPLOYEE_ID",
...
"DefaultCCSID":37,
"DblDefaultCCSID":0,
"LastChange":"2024-04-13-08.28.30.852000"
} ],
"Output Field Tables": [
{"GroupId": "DEFAULT",
"OutputTargetName": "Adabas_zos",
"OutputTableName": "T00011_00011",
"TableSequence": 1,
"TableName": "ADABAS_AI",
"ValidFrom":"2023-04-12-12.34.21.853000",
"ValidUntil":"9999-99-99-99.99.99.999999",
"MaxOccurs": 2,
"ActOccursField": "ADABAS_AI_C",
"SubTableOf": "T00011_00011"
} ],
"IDMS Path Data": [
{"GroupId": "DEFAULT",
"OutputTargetName": "IDMS_TARGET",
"OutputTableName": "DENTAL_CLAIMS",
"ValidFrom":"2024-01-23-21.57.12.654000",
"ValidUntil":"9999-99-99-99.99.99.999999",
"PathNumber": 1,
"PathSequence": 1,
"SetToNextMember": "EMP-COVERAGE",
"FieldsToNextMember": ""
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/OutputObjects"
HTTP/1.1 200 OK

{"Output tables inserted":[
{"GroupId":"DEFAULT","OutputTargetName":"DB9D_output","OutputTable":"DEMO.EMPLOYEES","ValidFro
m":"2024-02-01-11.18.36.432000","ValidUntil":"9999-99-99-99.99.99.999999"}
],"Output field tables inserted":[
{"GroupId":"DEFAULT","OutputTargetName":"Adabas_zos","OutputTableName":"T00011_00011","TableSequ
ence":1,"TableName":"ADABAS_AI","ValidFrom":"2023-04-12-12.34.21.853000","ValidUntil":"9999-99-9
9-99.99.99.999999"}
],"IDMS Path Data inserted":[
{"GroupId":"DEFAULT","OutputTargetName":"IDMS_TARGET","OutputTableName":"DENTAL_CLAIMS","PathNum
ber":1,"PathSequence":1,"ValidFrom":"2024-01-23-21.57.12.654000","ValidUntil":"9999-99-99-99.9
9.99.999999"}
]}
```

PUT OutputObjects -> Updating output tables

Calling this method allows to update an output table and other objects, related to output tables (OutputDMSPath and OutputFieldTables), in the repository.

To update, the key values of the object to be updated as well as all field values of the resulting object must be saved in a file in JSON format, and this file is then sent in the request data. The names of the field values correspond to the field names in the corresponding repository tables, and are not case sensitive. The names of the key values of the object to be updated is preceded by the prefix “Before_”.

The individual objects are grouped into JSON arrays. The identifiers “OutputTables“, “Output Field Tables“ and “IDMS Path Data“ are available for these groups, and are also not case sensitive.

Note: When key values of an output table are updated, all repository objects related to the output table through these key values (`OutputIDMSPaths`, `OutputFieldTables`, `OutputFields`, `ReplicationRules` and `ReplicationLinkages`) are also automatically updated so that the relationship of these objects to one another is not lost. It is therefore not necessary to additionally update these dependent objects.

A table in the repository table OutputTables is to be updated. To do this, a JSON file “data.json” with the following content (shown in abbreviated form) is created.

```
{"OutputTables":[
{"Before_GroupId":"DEFAULT",
"Before_OutputTargetName":"DB9D_output",
"Before_OutputTable":"DEMO.CUSTOMERS",
"Before_ValidFrom":"2024-04-15-10.06.59.521000",
"Before_ValidUntil":"9999-99-99-99.99.99.999999",
"GroupId":"DEFAULT",
"OutputTargetName":"DB9D_output",
"OutputTable":"PROD.CUSTOMERS",
"ValidFrom":"2024-04-15-10.06.59.521000",
"ValidUntil":"9999-99-99-99.99.99.999999",
"Description":"Changed creator",
...
"DefaultCCSID":37,
"DblDefaultCCSID":0,
"LastChange":"2024-05-13-09.17.54.734000"
}
]}
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/OutputObjects"
HTTP/1.1 200 OK

{"Output tables updated":[
{"GroupId":"DEFAULT","OutputTargetName":"DB9D_output","OutputTable":"DEMO.CUSTOMERS","ValidFro
m":"2024-04-15-10.06.59.521000","ValidUntil":"9999-99-99-99.99.99.999999"}
]}
```

A table in the repository table `OutputTables` is to be updated. At the same time, the related tables `OutputIDMSPaths` and `OutputFieldTables` should be updated. To do this, a JSON file “data.json” with the following content (shown in abbreviated form) is created.

```
{"OutputTables":[
{"Before_GroupId":"DEFAULT",
"Before_OutputTargetName":"DB9D_output",
"Before_OutputTable":"DEMO.EMPLOYEES",
"Before_ValidFrom":"2024-02-01-11.18.36.432000",
"Before_ValidUntil":"9999-99-99-99.99.99.999999",
"GroupId":"DEFAULT",
"OutputTargetName":"DB9D output",
"OutputTable":"DEMO.ANGESTELLTE",
"ValidFrom":"2024-02-01-11.18.36.432000",
"ValidUntil":"9999-99-99-99.99.99.999999",
"Description":"",
...
"DblDefaultCCSID":0,
"LastChange":"2024-04-13-08.28.30.852000"
} ],
"Output Field Tables": [
{"Before_GroupId": "DEFAULT",
"Before_OutputTargetName": "Adabas_zos",
"Before_OutputTableName": "T00011_00011",
"Before_TableSequence": 1,
"Before_TableName": "ADABAS_AI",
"Before_ValidFrom":"2023-04-12-12.34.21.853000",
"Before_ValidUntil":"9999-99-99-99.99.99.999999",
"GroupId": "DEFAULT",
"OutputTargetName": "Adabas_zos",
"OutputTableName": "T00022_00022",
"TableSequence": 1,
"TableName": "ADABAS_AJ",
"ValidFrom":"2023-04-12-12.34.21.853000",
"ValidUntil":"9999-99-99-99.99.99.999999",
"MaxOccurs": 2,
"ActOccursField": "ADABAS_AJ_C",
"SubTableOf": "T00022_00022"
} ],
"IDMS Path Data": [
{"Before_GroupId": "DEFAULT",
"Before_OutputTargetName": "IDMS_TARGET",
"Before_OutputTableName": "DENTAL_CLAIMS",
"Before_PathNumber": 1,
"Before_PathSequence": 1,
"Before_ValidFrom":"2024-01-23-21.57.12.654000",
"Before_ValidUntil":"9999-99-99-99.99.99.999999",
"GroupId": "DEFAULT",
"OutputTargetName": "IDMS_TARGET",
"OutputTableName": "DENTAL_BONUS",
"ValidFrom":"2024-01-23-21.57.12.654000",
"ValidUntil":"9999-99-99-99.99.99.999999",
"PathNumber": 3,
"PathSequence": 4,
"SetToNextMember": "EMP-BONUS",
"FieldsToNextMember": ""
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/OutputObjects"
HTTP/1.1 200 OK

{"Output tables updated":[
{"GroupId":"DEFAULT","OutputTargetName":"DB9D_output","OutputTable":"DEMO.EMPLOYEES","ValidFro
m":"2024-02-01-11.18.36.432000","ValidUntil":"9999-99-99-99.99.99.999999"}
],"Output field tables updated":[
{"GroupId":"DEFAULT","OutputTargetName":"Adabas_zos","OutputTableName":"T00011_00011","TableSequ
ence":1,"TableName":"ADABAS_AI","ValidFrom":"2023-04-12-12.34.21.853000","ValidUntil":"9999-99-9
9-99.99.99.999999"}
],"IDMS Path Data updated":[
{"GroupId":"DEFAULT","OutputTargetName":"IDMS_TARGET","OutputTableName":"DENTAL_CLAIMS","PathNum
ber":1,"PathSequence":1,"ValidFrom":"2024-01-23-21.57.12.654000","ValidUntil":"9999-99-99-99.9
9.99.999999"}
]}
```

DELETE OutputObjects -> Deleting output tables

Calling this method allows to delete an output table and other objects, related to output tables (OutputIDMSPaths and OutputFieldTables), from the repository.

To delete, the key values of the object to be deleted must be saved to a file in JSON format, and this file is then sent in the request data. The names of the key values correspond to the field names in the corresponding repository tables, and are not case sensitive.

The individual objects are grouped into JSON arrays. The identifiers “OutputTables”, “Output Field Tables” and “IDMS Path Data” are available for these groups, and are also not case sensitive.

Note: When an output table is deleted, all repository objects related to the output table (OutputIDMSPaths, OutputFieldTables, OutputFields, ReplicationRules and ReplicationLinkages) are also automatically deleted, since the existence of these objects makes no sense without the output table. It is therefore not necessary to additionally delete these dependent objects.

A table is to be deleted from the repository table OutputTables. To do this, a JSON file “data.json” with the following content is created.

```
{"OutputTables":[
{"GroupId":"DEFAULT",
"OutputTargetName":"DB9D_output",
"OutputTable":"DEMO.CUSTOMERS",
"ValidFrom":"2024-04-15-10.06.59.521000",
"ValidUntil":"9999-99-99-99.99.99.999999",
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX DELETE "http://127.0.0.1:8080/OutputObjects"
HTTP/1.1 200 OK

{"Output tables deleted":[
{"GroupId":"DEFAULT","OutputTargetName":"DB9D_output","OutputTable":"DEMO.CUSTOMERS","ValidFro
m":"2024-04-15-10.06.59.521000","ValidUntil":"9999-99-99-99.99.99.999999"}
]}
```

Entries should be deleted from the OutputIDMSPaths and OutputFieldTables tables. To do this, a JSON file “data.json” with the following content is created.

```
{
"Output Field Tables": [
{"GroupId": "DEFAULT",
"OutputTargetName": "Adabas_zos",
"OutputTableName": "T00022_00022",
"TableSequence": 1,
"TableName": "ADABAS_AJ",
"ValidFrom":"2023-04-12-12.34.21.853000",
"ValidUntil":"9999-99-99-99.99.99.999999"
}
],
"IDMS Path Data": [
{"GroupId": "DEFAULT",
"OutputTargetName": "IDMS_TARGET",
"OutputTableName": "DENTAL_BONUS",
"ValidFrom":"2024-01-23-21.57.12.654000",
"ValidUntil":"9999-99-99-99.99.99.999999",
"PathNumber": 3,
"PathSequence": 4
}
]
}
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX DELETE "http://127.0.0.1:8080/InputObjects"
HTTP/1.1 200 OK

{"Output field tables deleted":[
{"GroupId":"DEFAULT","OutputTargetName":"Adabas_zos","OutputTableName":"T00022_00022","TableSequ
ence":1,"TableName":"ADABAS_AJ","ValidFrom":"2023-04-12-12.34.21.853000","ValidUntil":"9999-99-9
9-99.99.99.999999"}
],"IDMS Path Data deleted":[
{"GroupId":"DEFAULT","OutputTargetName":"IDMS_TARGET","OutputTableName":"DENTAL_BONUS","PathNumb
er":3,"PathSequence":4,"ValidFrom":"2024-01-23-21.57.12.654000","ValidUntil":"9999-99-99-99.99.9
9.999999"}
]}
```

Error messages for resource OutputObjects

If an error occurs when executing an option of resource OutputObjects, an HTTP return code not equal to 200 is returned.

Below is a brief overview of possible error messages and their causes:

```
curl -H "@auth" -iX GETT http://127.0.0.1:8080/OutputObjects
HTTP/1.1 400 Bad Request

{"Error": "Method 'GETT' is not supported"}
```

An incorrect method was used in the call. The call needs to be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputObjects?groupid=MyGroup
HTTP/1.1 400 Bad Request

{"Error": "OutputObjects failed",
"Reason": "Specified groupid 'MyGroup' is not defined in repository."}
```

A group was used in the call that is not defined in the repository. The group to be used must be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputObjects?
outputtargetname=MyTarget
HTTP/1.1 400 Bad Request

{"Error": "OutputObjects failed",
"Reason": "Specified Output Target name 'DEFAULT.MyTarget' is not defined in repository."}
```

An Output Target was used in the call that is not defined in the repository. The Output Target to be used needs to be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputObjects?
validfrom=24-03-01-12.34.56.123456
HTTP/1.1 400 Bad Request

{"Error": "OutputObjects failed",
"Reason": "Specified timestamp '24-03-01-12.34.56.123456' has invalid format. Valid format is 'Y
YYY-MM-DD-HH.MM.SS.FFFFFF'"}
```

A timestamp with an invalid format was used in the call. The timestamp to be used must be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputObjects ...
HTTP/1.1 500 Internal Server Error

{"Error": "OutputObject failed",
"Reason": "Caught exception while creating result",
"Exception": "<Exception message>"}
```

If there is a problem with the internal processing of a call, this message will be returned. The placeholder <Exception message> is replaced by a message that corresponds to the problem.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/OutputObjects"
HTTP/1.1 500 Internal Server Error

{"Error":"OutputObjects failed",
"Reason":"Got SQL error message while inserting ",
"Object":"OutputTable DEMO.CUSTOMERS (2024-04-15-10.06.59.521000 – 9999-99-99-99.99.99.999999)",
"SQL message":"TCS0597E;70R0,WIN64_6.2.9200; Repository SQL error: TCS0256E;70R0,WIN64_6.2.9200;
PostgreSQL libpq SQLSTATE 23505, ERROR: duplicate key value violates unique constraint »outputta
bles_pkey« DETAIL: Key »(groupid, sourcetype, sourcename, validfrom, validuntil)=(DEFAULT , DB9
D_output , DEMO.CUSTOMERS , 2024-04-15-10.06.59.521000, 9999-99-99-99.99.99.999999)« already exi
sts.; N/A."}
```

If there are problems (SQL errors) when inserting, updating or deleting objects in the repository, this message is returned. The affected object is given as well as the error message received from the database.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/OutputObjects"
HTTP/1.1 400 Bad Request

{"Error":"OutputObjects failed",
"Reason":"Updating 'OutputIDMSPath' object in repository not possible, because not all key value
s of 'Before' object have been specified.",
"Object":"OutputIDMSPath RESTAPI.IDMS_TARGET.DENTAL_CLAIMS.0.0 (2024-01-23-21.57.12.654000 - 999
9-99-99-99.99.99.999999)"}
```

If not all key values are provided to uniquely identify an object, this will be indicated with a message like this.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/OutputObjects"
HTTP/1.1 400 Bad Request

{"Error":"OutputObjects failed",
"Reason":"Updating 'OutputIDMSPath' object in repository not possible, because object does not e
xist in repository.",
"Object":"OutputIDMSPath RESTAPI.IDMS_TARGET.DENTAL_CLAIMS.2.2 (2024-01-23-21.57.12.654000 - 999
9-99-99-99.99.99.999999)"}
```

If no object matching the key values passed is found in the repository, this will be indicated with a message like this.

Resource InputFields (Repository input fields)

Using the resource InputFields repository definitions of input fields can be retrieved.

OPTIONS InputFields -> Method Overview

```
curl -H "@auth" -iX OPTIONS http://127.0.0.1:8080/InputFields
HTTP/1.1 200 OK

{"= HTTP method =              ":"= Explanation =",
"GET    /InputFields          ":"Get information for input field(s)",
"POST   /InputFields          ":"Add a new or replace an input field",
"PUT    /InputFields          ":"Change an input field",
"DELETE /InputFields          ":"Delete an input field",
"= Optional queries for GET = ":"= Explanation =",
"groupid=<GroupId>              ":"Return fields for group <GroupId> only (default value = 'DEFA
ULT')",
"sourcetype=<SourceType>      ":"Return fields for type <SourceType> only",
"sourcename=<SourceName>      ":"Return fields for table <SourceName> only",
"fieldname=<FieldName>        ":"Return fields with name <FieldName> only",
"validfrom=<ValidFrom>        ":"Return fields valid at <ValidFrom> only",
"details=<Y|N>                ":"Returned result: 'N' key fields only,
 'Y' all fields (default value = 'N')",
"updatesequencenumbers=<Y|N>  ":"Updating the seq. nr. of following fields: 'N' don't update,
 'Y' do update (default value = 'Y')",
"= Request data =             ":"= Explanation =",
"Data in JSON format          ":"Use data to specify the attributes of the fields(s) to insert/u
pdate/delete"}
```

GET InputFields -> Information about input fields

The simple call without further query parameters returns all input fields of the 'DEFAULT' (standard) group as a result.

The result is returned in JSON notation. It informs whether the output is with all details or not and the individual input fields as JSON objects within a JSON array called ‘Results’. The key names of the key-value pairs in these objects correspond to the names of the fields of the repository objects (here, for example, the field names 'GroupId', 'SourceName', ‚FieldName‘ and 'ValidFrom' of the repository table 'InputFields').

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/InputFields
HTTP/1.1 200 OK

{"Details": false,
"Results": [
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00001.00001",
"FieldSequence": 1,
"FieldName": "ISN",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
...
{"GroupId": "DEFAULT",
"SourceType": "VSAM",
"SourceName": "VSAM.KVBEST.FILE",
"FieldSequence": 121,
"FieldName": "FILLER",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

The call can be restricted by specifying additional parameters in the query data. Only the fields that match the criteria of the parameters are returned.

The following parameters are available:

Parameter	Description
groupid	Restricts the determined fields to a group (the default is 'DEFAULT')
sourcetype	Restricts the determined fields to a source type
sourcename	Restricts the determined fields to a table name
fieldname	Restricts the determined fields to a field name
validform	Restricts the determined fields to a timestamp at which the field must be valid
details	Used to set the level of detail of the return. The possible values are: Y: all fields of the repository object N: only the key fields of the repository object (default)

Note:

- If all restrictive parameters ('groupid', 'sourcetype', 'sourcename', ‚fieldname‘ and 'validfrom') are set, 'Y' is automatically used for 'details'. In this case the result only consists of one input field with all the information from the repository definition of this field.
- The meaning of “all the information from the repository definition” is the information that is relevant to the corresponding field. For example, for a field of a Db2 table, all type-specific information that is not related to Db2 (e.g. the repository fields ‚AdabasName‘, ‚BigDataNr1‘, ...) will be suppressed.
- If these parameters contain special characters such as spaces, <, >, ^, $, etc., they must be encoded accordingly before sending to the REST API server. A table with some characters and their encoding can be found in chapter „Resource Import“.

Examples of using the parameters

Only the fields of Adabas tables should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/InputFields?sourcetype=ADABAS
HTTP/1.1 200 OK

{"Details": false,
"Results": [
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00001.00001",
"FieldSequence": 1,
"FieldName": "ISN",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"},
...
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "01111.01111",
"FieldSequence": 29,
"FieldName": "ADABAS_AZ",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

Only the fields of the Oracle table “ORCL.DEMO.ARTIKEL” should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/InputFields?sourcetype=ORACLE&sourcename=ORCL.DEM
O.ARTIKEL
HTTP/1.1 200 OK

{"Details": false,
"Results": [
{"GroupId": "DEFAULT",
"SourceType": "ORACLE",
"SourceName": "ORCL.DEMO.ARTIKEL",
"FieldSequence": 1,
"FieldName": "ID",
"ValidFrom": "2019-11-25-14.29.45.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"},
...
{"GroupId": "DEFAULT",
"SourceType": "ORACLE",
"SourceName": "ORCL.DEMO.ARTIKEL",
"FieldSequence": 7,
"FieldName": "STOCK",
"ValidFrom": "2019-11-25-14.29.45.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

Only the field “STOCK” of the Oracle table “ORCL.TEST.ARTIKEL” should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/InputFields?sourcetype=ORACLE&sourcename=ORCL.TES
T.ARTIKEL&fieldname=STOCK
HTTP/1.1 200 OK

{"Details": true,
"Results": [
{"GroupId": "DEFAULT",
"SourceType": "ORACLE",
"SourceName": "ORCL.TEST.ARTIKEL",
"FieldSequence": 7,
"FieldName": "BESTAND",
"ValidFrom": "2020-07-15-08.29.21.000000",
"ValidUntil": "2021-07-15-08.29.20.999999"
},
{"GroupId": "DEFAULT",
"SourceType": "ORACLE",
"SourceName": "ORCL.TEST.ARTIKEL",
"FieldSequence": 7,
"FieldName": "BESTAND",
"ValidFrom": "2021-07-15-08.29.21.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

Note: The table was reimported, which is why two versions of the table and therefore also of the field exist with different validity ranges.

Only the field “STOCK” of the Oracle table “ORCL.TEST.ARTIKEL”, valid at time “2021-07-15-08.29.21.000000”, should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/InputFields?sourcetype=ORACLE&sourcename=ORCL.TES
T.ARTIKEL&fieldname=STOCK&
validfrom=2021-07-15-08.29.21.000000
HTTP/1.1 200 OK

{"Details": true,
"Results": [
{"GroupId": "DEFAULT",
"SourceType": "ORACLE",
"SourceName": "ORCL.TEST.ARTIKEL",
"FieldSequence": 7,
"FieldName": "BESTAND",
"ValidFrom": "2021-07-15-08.29.21.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
"RedefineOfField": "",
"RedefineSequence": 0,
"FieldActive": "Y",
"FieldType": 3005,
"FieldLen": 22,
"FieldPrecision": 10,
"FieldScale": 0,
"FieldCCSID": 0,
"ExitFlag": 0,
"UserFieldExitNo": 0,
"SysFieldExitNo": 0,
"WithInTable": "",
"DBInternalSegCol": 7,
"DefaultValue": "",
"DefaultValueType": "",
"SpecialName": ""}]}
```

Note: By defining all other parameters, the parameter 'details=Y' is automatically set and the output contains all relevant information about the field.

POST InputFields -> Inserting input fields

The call allows input fields to be inserted into the repository.

For the insert all field values of the objects must be stored in a file in JSON format, and this file is then sent in the request data. The names of the field values correspond to the field names in repository table “InputFields”, and are not case sensitive.

The individual input field objects are embedded within the JSON array “InputFields”, this identifier is also not case sensitive.

Note: Input fields have a position (sequence number) within a table. When a field is inserted, the position number is defined along with the values used to define the field. However, if a field is not added to the end of the table, but is inserted somewhere between existing fields, then this would result in fields with identical position numbers, which is not allowed. As a standard, the position numbers of the fields following the inserted field are automatically increased, so that an ascending order of the field positions is again guaranteed. If this behavior is not desired, this can be achieved by setting the parameter “`updatesequencenumbers=n`” in the query data.

Two fields are to be inserted into the repository table InputFields. To do this, a JSON file “data.json” with the following content (shown in abbreviated form) is created.

```
{
"InputFields": [
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00010.00010",
"FieldSequence": 34,
"FieldName": "ADABAS_ZA",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
...
"AdabasName": "ZA"
},
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00010.00010",
"FieldSequence": 35,
"FieldName": "ADABAS_ZB",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
...
"AdabasName": "ZB"
}
]
}
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/InputFields"
HTTP/1.1 200 OK

{"UpdateSequenceNumbers":true,"Input fields inserted":[
{"GroupId":"DEFAULT","SourceType":"ADABAS","SourceName":"00010.00010","FieldSequence":34,"FieldN
ame":"ADABAS_ZA","ValidFrom":"0000-00-00-00.00.00.000000","ValidUntil":"9999-99-99-99.99.99.9999
99"},
{"GroupId":"DEFAULT","SourceType":"ADABAS","SourceName":"00010.00010","FieldSequence":35,"FieldN
ame":"ADABAS_ZB","ValidFrom":"0000-00-00-00.00.00.000000","ValidUntil":"9999-99-99-99.99.99.9999
99"}
]}
```

PUT InputFields -> Updating output fields

Calling this method allows to update input fields in the repository.

To update, the key values of the object to be updated as well as all field values of the resulting object must be saved in a file in JSON format, and this file is then sent in the request data. The names of the field values correspond to the field names in repository table “InputFields”, and are not case sensitive. The names of the key values of the object to be updated is preceded by the prefix “Before_”.

The individual input field objects are embedded within the JSON array “InputFields”, this identifier is also not case sensitive.

Note: Input fields have a position (sequence number) within a table. When a field is updated, this position number should not be changed in order not to destroy the numbering of the fields within a table.

A field is to be updated in the repository table InputFields. To do this, a JSON file “data.json” with the following content (shown in abbreviated form) is created.

```
{
"InputFields": [
{"Before_GroupId": "DEFAULT",
"Before_SourceType": "ADABAS",
"Before_SourceName": "00010.00010",
"Before_FieldSequence": 35,
"Before_FieldName": "ADABAS_ZB",
"Before_ValidFrom": "0000-00-00-00.00.00.000000",
"Before_ValidUntil": "9999-99-99-99.99.99.999999",
"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00010.00010",
"FieldSequence": 2,
"FieldName": "ADABAS_ZB",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "Changing the type of the field",
"RedefineOfField": "",
"RedefineSequence": 0,
"FieldActive": "Yes",
"FieldType": 2500,
"FieldLen": 2,
...
"AdabasName": "AA"
}
]
}
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/InputFields?updatesequencenumber
s=n"
HTTP/1.1 200 OK

{"UpdateSequenceNumbers":false,"Input fields updated":[
{"GroupId":"DEFAULT","SourceType":"ADABAS","SourceName":"00010.00010","FieldSequence":2,"FieldNa
me":"ADABAS_AA","ValidFrom":"0000-00-00-00.00.00.000000","ValidUntil":"9999-99-99-99.99.99.99999
9"}
]}
```

DELETE InputFields -> Deleting input fields

Calling this method allows to delete input fields from the repository.

To delete, the key values of the object to be deleted must be saved in a file in JSON format, and this file is then sent in the request data. The names of the field values correspond to the field names in repository table “InputFields”, and are not case sensitive.

The individual input field objects are embedded within the JSON array “InputFields”, this identifier is also not case sensitive.

Note: Input fields have a position (sequence number) within a table. If a field is deleted, a gap will be created in the numbering of the fields within the table, which should not be the case. As a standard, the position numbers of the fields following the deleted field are automatically reduced, so that a consistent, ascending order of the field positions is guaranteed again. If this behavior is not desired, this can be achieved by setting the parameter “updatesequencenumbers=n” in the query data.

A field is to be deleted from the repository table InputFields. To do this, a JSON file “data.json” with the following content is created.

```
{
"InputFields": [
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00010.00010",
"FieldSequence": 1,
"FieldName": "ISN",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"
}
]
}
```

This file is then passed in the call.

```
{"UpdateSequenceNumbers":true,"Input fields deleted":[
{"GroupId":"DEFAULT","SourceType":"ADABAS","SourceName":"00010.00010","FieldSequence":1,"FieldNa
me":"ISN","ValidFrom":"0000-00-00-00.00.00.000000","ValidUntil":"9999-99-99-99.99.99.999999"}
]}curl -d @data.json -H "@auth" -iX DELETE "http://127.0.0.1:8080/InputFields"
```

Two fields are to be deleted from the repository table InputFields. To do this, a JSON file “data.json” with the following content is created.

Important: In order to ensure the correct numbering of the remaining fields in the table, the position numbers of the fields following the field to be deleted are corrected. Therefore, when deleting multiple fields within a table, it is important to specify the fields in descending order. Otherwise, after deleting position 1, position 2 to be deleted would itself be position 1 and deleting the second field would fail due to an object not being found in the repository.

```
{
"InputFields": [
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00010.00010",
"FieldSequence": 2,
"FieldName": "ADABAS_AA",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"
},
{"GroupId": "DEFAULT",
"SourceType": "ADABAS",
"SourceName": "00010.00010",
"FieldSequence": 1,
"FieldName": "ISN",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"
}
]
}
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX DELETE "http://127.0.0.1:8080/InputFields"
HTTP/1.1 200 OK

{"UpdateSequenceNumbers":true,"Input fields deleted":[
{"GroupId":"DEFAULT","SourceType":"ADABAS","SourceName":"00010.00010","FieldSequence":2,"FieldNa
me":"ADABAS_AA","ValidFrom":"0000-00-00-00.00.00.000000","ValidUntil":"9999-99-99-99.99.99.99999
9"},
{"GroupId":"DEFAULT","SourceType":"ADABAS","SourceName":"00010.00010","FieldSequence":1,"FieldNa
me":"ISN","ValidFrom":"0000-00-00-00.00.00.000000","ValidUntil":"9999-99-99-99.99.99.999999"}
]}
```

Error messages for resource InputFields

If an error occurs when executing an option of resource InputFields, an HTTP return code not equal to 200 is returned.

Below is a brief overview of possible error messages and their causes:

```
curl -H "@auth" -iX GETT http://127.0.0.1:8080/InputFields
HTTP/1.1 400 Bad Request

{"Error": "Method 'GETT' is not supported"}
```

An incorrect method was used in the call. The call needs to be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/InputFields?groupid=MyGroup
HTTP/1.1 400 Bad Request

{"Error": "InputFields failed",
"Reason": "Specified groupid 'MyGroup' is not defined in repository."}
```

A group was used in the call that is not defined in the repository. The group to be used must be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/InputFields?
validfrom=24-03-01-12.34.56.123456
HTTP/1.1 400 Bad Request

{"Error": "InputFields failed",
"Reason": "Specified timestamp '24-03-01-12.34.56.123456' has invalid format. Valid format is 'Y
YYY-MM-DD-HH.MM.SS.FFFFFF'"}
```

A timestamp with an invalid format was used in the call. The timestamp to be used must be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/InputFields ...
HTTP/1.1 500 Internal Server Error

{"Error": "InputFields failed",
"Reason": "Caught exception while creating result",
"Exception": "<Exception message>"}
```

If there is a problem with the internal processing of a call, this message will be returned. The placeholder <Exception message> is replaced by a message that corresponds to the problem.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/InputFields"
HTTP/1.1 500 Internal Server Error

{"Error":"InputFields failed",
"Reason":"Got SQL error message while inserting ",
"Object":"InputField DEFAULT.ADABAS.00010.00010.ISN (0000-00-00-00.00.00.000000 – 9999-99-99-9
9.99.99.999999)",
"SQL message":"TCS0597E;70R0,WIN64_6.2.9200; Repository SQL error: TCS0256E;70R0,WIN64_6.2.9200;
PostgreSQL libpq SQLSTATE 23505, ERROR: duplicate key value violates unique constraint »inputfie
lds_pkey« DETAIL: key »(groupid, sourcetype, sourcename, fieldsequence, fieldname, sequencenumbe
r, validfrom, validuntil)=(DEFAULT , ADABAS , 00001.00001 , 1 , ISN, 0000-00-00-00.00.00.000000,
9999-99-99-99.99.99.999999)« already exists.; N/A."}
```

If there are problems (SQL errors) when inserting, updating or deleting objects in the repository, this message is returned. The affected object is given as well as the error message received from the database.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/InputFields"
HTTP/1.1 400 Bad Request

{"Error":"InputFields failed",
"Reason":"Updating 'InputField' object in repository not possible, because not all key values of
'Before' object have been specified.",
"Object":"InputField DEFAULT.ADABAS.00010.00010.ISN ( 0000-00-00-00.00.00.000000 - )"}
```

If not all key values are provided to uniquely identify an object, this will be indicated with a message like this.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/InputFields"
HTTP/1.1 400 Bad Request

{"Error":"InputFields failed",
"Reason":"Updating 'InputField' object in repository not possible, because object does not exist
in repository.",
"Object":" InputTable DEFAULT.ADABAS.00010.00010.USN (0000-00-00-00.00.00.000000 - 9999-99-99-9
9.99.99.999999)"}
```

If no object matching the key values passed is found in the repository, this will be indicated with a message like this.

Resource OutputFields (Repository output fields)

OPTIONS OutputFields -> Method Overview

```
curl -H "@auth" -iX OPTIONS http://127.0.0.1:8080/OutputFields
{"= HTTP method =               ":"= Explanation =",
"GET /OutputFields              ":"Get information for output field(s)",
"POST   /OutputFields           ":"Add a new or replace an output field",
"PUT    /OutputFields           ":"Change an output field",
"DELETE /OutputFields           ":"Delete an output field",
"= Optional queries for GET =  ":"= Explanation =",
"groupid=<GroupId>                ":"Return fields for group <GroupId> only (default value = 'DEF
AULT')",
"outputtargetname=<TargetName> ":"Return fields for target <TargetName> only",
"outputtablename=<OutputTable> ":"Return fields for table <OutputTable> only",
"outputfieldname=<FieldName>   ":"Return fields with name <FieldName> only",
"validfrom=<ValidFrom>          ":"Return fields valid at <ValidFrom> only",
"details=<Y|N>                  ":"Returned result: 'N' key fields only,
 'Y' all fields (default value = 'N')",
"updatesequencenumbers=<Y|N>   ":"Updating the seq. nr. of following fields: 'N' don't update,
 'Y' do update (default value = 'Y')",
"= Request data =               ":"= Explanation =",
"Data in JSON format            ":"Use data to specify the attributes of the fields(s) to insert/
update/delete"}
```

GET OutputFields -> Information about output fields

The simple call without further query parameters returns all output fields of the 'DEFAULT' (standard) group as a result.

The result is returned in JSON notation. It informs whether the output is with all details or not and the individual output fields as JSON objects within a JSON array called ‘Results’. The key names of the key-value pairs in these objects correspond to the names of the fields of the repository objects (here, for example, the field names 'GroupId', ‚OutputTableName‘, ‚OutputFieldName‘ and ‚ValidFrom‘ of the repository table 'OutputFields').

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputFields
HTTP/1.1 200 OK

{"Details": false,
"Results": [
{"GroupId": "DEFAULT",
"OutputTargetName": "Snowflake_DB",
"OutputTableName": "PUBLIC.ZWOELF_ALLTYPES",
"OutputFieldSeq": 1,
"OutputFieldName": "ZEICHEN",
"ValidFrom": "2022-04-22-13.15.20.500000",
"ValidUntil": "9999-99-99-99.99.99.999999"},
...
{"GroupId": "DEFAULT",
"OutputTargetName": "Target_File",
"OutputTableName": "demo.ddltest",
"OutputFieldSeq": 4,
"OutputFieldName": "lastfield",
"ValidFrom": "2023-07-17-12.19.22.500000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

The call can be restricted by specifying additional parameters in the query data. Only the fields that match the criteria of the parameters are returned.

The following parameters are available:

Parameter	Description
groupid	Restricts the determined fields to a group (the default is 'DEFAULT')
outputtargetname	Restricts the determined fields to an Output Target
outputtablename	Restricts the determined fields to a table name
outputfieldname	Restricts the determined fields to a field name
validform	Restricts the determined fields to a timestamp at which the field must be valid
details	Used to set the level of detail of the return. The possible values are: Y: all fields of the repository object N: only the key fields of the repository object (default)

Note:

- If all restrictive parameters ('groupid', 'outputtargetname', 'outputtablename', ‚outputfieldname‘ and 'validfrom') are set, 'Y' is automatically used for 'details'. In this case the result only consists of one output field with all the information from the repository definition of this field.
- The meaning of "all the information from the repository definition" is the information that is relevant to the corresponding field. For example, for a field of an Adabas table, all type-specific information that is not related to Adabas (e.g. the repository fields ‚IDMSControlField‘, ‚IDMSOffsetInOwner‘, ...) will be suppressed.
- If these parameters contain special characters such as spaces, <, >, ^, $, etc., they must be encoded accordingly before sending to the REST API server. A table with some characters and their encoding can be found in chapter „Resource Import".

Examples of using the parameters

Only the fields of the Output Target "Target_Postgres" should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputFields?outputtargetname=Target_Postgres
```

```
HTTP/1.1 200 OK

{"Details": false,
"Results": [
{"GroupId": "DEFAULT",
"OutputTargetName": "Target_Postgres",
"OutputTableName": "DEMO.ARTIKEL",
"OutputFieldSeq": 1,
"OutputFieldName": "ID",
"ValidFrom": "2021-01-18-13.39.23.500000",
"ValidUntil": "9999-99-99-99.99.99.999999"},
...
{"GroupId": "DEFAULT",
"OutputTargetName": "Target_Postgres",
"OutputTableName": "TEST.TYPES",
"OutputFieldSeq": 12,
"OutputFieldName": "LAST",
"ValidFrom": "2021-07-09-07.14.57.727000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

Only the fields of table “public.artikel” of Output Target “Target_Postgres” should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputFields?outputtargetname=Target_Postgres&outp
uttablename=public.artikel
HTTP/1.1 200 OK

{"Details": false,
"Results": [
{"GroupId": "DEFAULT",
"OutputTargetName": "Target_Postgres",
"OutputTableName": "public.artikel",
"OutputFieldSeq": 1,
"OutputFieldName": "ID",
"ValidFrom": "2022-04-27-12.20.07.306000",
"ValidUntil": "9999-99-99-99.99.99.999999"},
...
{"GroupId": "DEFAULT",
"OutputTargetName": "Target_Postgres",
"OutputTableName": "public.artikel",
"OutputFieldSeq": 7,
"OutputFieldName": "BESTAND",
"ValidFrom": "2022-04-27-12.20.07.306000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

Only the field “PRICE” of table “test.artikel” of Output Target “Target_Postgres” should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputFields?outputtargetname=Target_Postgres&outp
uttablename=test.artikel&
outputfieldname=PRICE
HTTP/1.1 200 OK

{"Details": false,
"Results": [
{"GroupId": "DEFAULT",
"OutputTargetName": "Target_Postgres",
"OutputTableName": "test.artikel",
"OutputFieldSeq": 5,
"OutputFieldName": "PREIS",
"ValidFrom": "2022-04-27-12.20.07.306000",
"ValidUntil": "2023-06-13-17.31.26.499999"
},
{"GroupId": "DEFAULT",
"OutputTargetName": "Target_Postgres",
"OutputTableName": "test.artikel ",
"OutputFieldSeq": 5,
"OutputFieldName": "PREIS",
"ValidFrom": "2023-06-13-17.31.26.500000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

Note: The table was reimported, which is why two versions of the table and therefore also of the field exist with different validity ranges.

Only the field “PRICE” of table “test.artikel” of Output Target “Target_Postgres”, valid at time “2023-06-13-17.31.26.500000”, should be returned.

```
curl -H "@auth" -iX GET ht tp://127.0.0.1:8080/OutputFields?outputtargetname=Target_Postgres&out
puttablename=test.artikel&
outputfieldname=PRICE&validfrom=2023-06-13-17.31.26.500000
HTTP/1.1 200 OK

{"Details": true,
"Results": [
{"GroupId": "DEFAULT",
"OutputTargetName": "Target_Postgres",
"OutputTableName": "test.artikel",
"OutputFieldSeq": 5,
"OutputFieldName": "PREIS",
"ValidFrom": "2023-06-13-17.31.26.500000",
"ValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
"FieldActive": "Y",
"FieldType": 485,
"FieldLen": 10,
"FieldScale": 2,
"FieldCCSID": 0,
"UnsignedNumeric": "N",
"RedefineOfField": "",
"RedefineSequence": 0,
"ExitFlag": 0,
"UserFieldExitNo": 0,
"DefaultValue": "",
"NULLTestValue": "",
"WithInTable": "",
"ExcludeFromSupTest": "N"}]}
```

Note: By defining all other parameters, the parameter 'details=Y' is automatically set and the output contains all relevant information about the field.

POST OutputFields -> Inserting output fields

The call allows output fields to be inserted into the repository.

For the insert all field values of the objects must be stored in a file in JSON format, and this file is then sent in the request data. The names of the field values correspond to the field names in repository table “OutputFields”, and are not case sensitive.

The individual output field objects are embedded within the JSON array “OutputFields”, this identifier is also not case sensitive.

Note: Output fields have a position (sequence number) within a table. When a field is inserted, the position number is defined along with the values used to define the field. However, if a field is not added to the end of the table, but is inserted somewhere between existing fields, then this would result in fields with identical position numbers, which is not allowed. As a standard, the position numbers of the fields following the inserted field are automatically increased, so that an ascending order of the field positions is again guaranteed. If this behavior is not desired, this can be achieved by setting the parameter “updatesequencenumbers=n” in the query data.

Two fields are to be inserted into the repository table OutputFields. To do this, a JSON file “data.json” with the following content (shown in abbreviated form) is created.

```
{
"OutputFields": [
{"GroupId":"DEFAULT",
"OutputTargetName":"Target_MsSQL",
"OutputTableName":"test.dollar",
"OutputFieldSeq":1,
"OutputFieldName":"nr",
"ValidFrom":"2024-04-11-11.39.47.955000",
"ValidUntil":"2024-05-14-05.33.24.045999",
"Description":"",
...
"ExcludeFromSupTest":"N"
},
{"GroupId":"DEFAULT",
"OutputTargetName":"Target_MsSQL",
"OutputTableName":"test.dollar",
"OutputFieldSeq":2,
"OutputFieldName":"data",
"ValidFrom":"2024-04-11-11.39.47.955000",
"ValidUntil":"2024-05-14-05.33.24.045999",
"Description":"",
...
"ExcludeFromSupTest":"N"
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/OutputFields"
HTTP/1.1 200 OK

{"UpdateSequenceNumbers":true,"Output fields inserted":[
{"GroupId":"DEAFULT","OutputTargetName":"Target_MsSQL","OutputTableName":"test.dollar","OutputFi
eldSeq":1,"OutputFieldName":"nr","ValidFrom":"2024-04-11-11.39.47.955000","ValidUntil":"2024-0
5-14-05.33.24.045999"},
{"GroupId":"DEAFULT","OutputTargetName":"Target_MsSQL","OutputTableName":"test.dollar","OutputFi
eldSeq":2,"OutputFieldName":"data","ValidFrom":"2024-04-11-11.39.47.955000","ValidUntil":"2024-0
5-14-05.33.24.045999"}
]}
```

PUT OutputFields -> Updating output fields

Calling this method allows to update output fields in the repository.

To update, the key values of the object to be updated as well as all field values of the resulting object must be saved in a file in JSON format, and this file is then sent in the request data. The names of the field values correspond to the field names in repository table “OutputFields“, and are not case sensitive. The names of the key values of the object to be updated is preceded by the prefix “Before_”.

The individual output field objects are embedded within the JSON array “OutputFields”, this identifier is also not case sensitive.

Note: Output fields have a position (sequence number) within a table. When a field is updated, this position number should not be changed in order not to destroy the numbering of the fields within a table.

A field is to be updated in the repository table OutputFields. To do this, a JSON file “data.json” with the following content (shown in abbreviated form) is created.

```
{
"OutputFields": [
{"Before_GroupId":"DEFAULT",
"Before_OutputTargetName":"Target_MsSQL",
"Before_OutputTableName":"test.dollar",
"Before_OutputFieldSeq":2,
"Before_OutputFieldName":"data",
"Before_ValidFrom":"2024-04-11-11.39.47.955000",
"Before_ValidUntil":"2024-05-14-05.33.24.045999",
"GroupId":"DEFAULT",
"OutputTargetName":"Target_MsSQL",
"OutputTableName":"test.dollar",
"OutputFieldSeq":2,
"OutputFieldName":"data",
"ValidFrom":"2024-04-11-11.39.47.955000",
"ValidUntil":"2024-05-14-05.33.24.045999",
"Description":"Changed type and length of field",
"FieldActive":"Y",
"FieldType":449,
"FieldLen":200,
...
"ExcludeFromSupTest":"N"
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/OutputFields?updatesequencenumber
s=n"
HTTP/1.1 200 OK

{"UpdateSequenceNumbers":true,"Output fields updated":[
{"GroupId":"DEFAULT","OutputTargetName":"Target_MsSQL","OutputTableName":"test.dollar","OutputFi
eldSeq":2,"OutputFieldName":"data","ValidFrom":"2024-04-11-11.39.47.955000","ValidUntil":"2024-0
5-14-05.33.24.045999"}
]}
```

DELETE OutputFields -> Deleting output fields

Calling this method allows to delete output fields from the repository.

To delete, the key values of the object to be deleted must be saved in a file in JSON format, and this file is then sent in the request data. The names of the field values correspond to the field names in repository table “OutputFields”, and are not case sensitive.

The individual output field objects are embedded within the JSON array “OutputFields”, this identifier is also not case sensitive.

Note: Output fields have a position (sequence number) within a table. If a field is deleted, a gap will be created in the numbering of the fields within the table, which should not be the case. As a standard, the position numbers of the fields following the deleted field are automatically reduced, so that a consistent, ascending order of the field positions is guaranteed again. If this behavior is not desired, this can be achieved by setting the parameter “updatesequencenumbers=n” in the query data.

A field is to be deleted from the repository table OutputFields. To do this, a JSON file “data.json” with the following content is created.

```
{
"OutputFields": [
{"GroupId":"DEFAULT",
"OutputTargetName":"Target_MsSQL",
"OutputTableName":"test.dollar",
"OutputFieldSeq":2,
"OutputFieldName":"data",
"ValidFrom":"2024-04-11-11.39.47.955000",
"ValidUntil":"2024-05-14-05.33.24.045999"
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX DELETE "http://127.0.0.1:8080/OutputFields"
{"UpdateSequenceNumbers":true,"Output fields deleted":[
{"GroupId":"DEFAULT","OutputTargetName":"Target_MsSQL","OutputTableName":"test.dollar","OutputFi
eldSeq":2,"OutputFieldName":"data","ValidFrom":"2024-04-11-11.39.47.955000","ValidUntil":"2024-0
5-14-05.33.24.045999"}
]}
```

Two fields are to be deleted from the repository table OutputFields. To do this, a JSON file “data.json” with the following content is created.

Important: In order to ensure the correct numbering of the remaining fields in the table, the position numbers of the fields following the field to be deleted are corrected. Therefore, when deleting multiple fields within a table, it is important to specify the fields in descending order. Otherwise, after deleting position 1, position 2 to be deleted would itself be position 1 and deleting the second field would fail due to an object not being found in the repository.

```
{
"OutputFields": [
{"GroupId":"DEFAULT",
"OutputTargetName":"Target_MsSQL",
"OutputTableName":"test.dollar",
"OutputFieldSeq":2,
"OutputFieldName":"data",
"ValidFrom":"2024-04-11-11.39.47.955000",
"ValidUntil":"2024-05-14-05.33.24.045999"
},
{"GroupId":"DEFAULT",
"OutputTargetName":"Target_MsSQL",
"OutputTableName":"test.dollar",
"OutputFieldSeq":1,
"OutputFieldName":"nr",
"ValidFrom":"2024-04-11-11.39.47.955000",
"ValidUntil":"2024-05-14-05.33.24.045999",
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX DELETE "http://127.0.0.1:8080/OutputFields"
HTTP/1.1 200 OK

{"UpdateSequenceNumbers":true,"Output fields deleted":[
{"GroupId":"DEFAULT","OutputTargetName":"Target_MsSQL","OutputTableName":"test.dollar","OutputFi
eldSeq":2,"OutputFieldName":"data","ValidFrom":"2024-04-11-11.39.47.955000","ValidUntil":"2024-0
5-14-05.33.24.045999"},
{"GroupId":"DEFAULT","OutputTargetName":"Target_MsSQL","OutputTableName":"test.dollar","OutputFi
eldSeq":1,"OutputFieldName":"nr","ValidFrom":"2024-04-11-11.39.47.955000","ValidUntil":"2024-0
5-14-05.33.24.045999"}
]}
```

Error messages for resource OutputFields

If an error occurs when executing an option of resource OutputFields, an HTTP return code not equal to 200 is returned.

Below is a brief overview of possible error messages and their causes:

```
curl -H "@auth" -iX GETT http://127.0.0.1:8080/OutputFields
HTTP/1.1 400 Bad Request

{"Error": "Method 'GETT' is not supported"}
```

An incorrect method was used in the call. The call needs to be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputFields?groupid=MyGroup
HTTP/1.1 400 Bad Request

{"Error": "OutputFields failed",
"Reason": "Specified groupid 'MyGroup' is not defined in repository."}
```

A group was used in the call that is not defined in the repository. The group to be used must be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputFields?
outputtargetname=MyTarget
HTTP/1.1 400 Bad Request

{"Error": "OutputFields failed",
"Reason": "Specified Output Target name 'DEFAULT.MyTarget' is not defined in repository."}
```

An Output Target was used in the call that is not defined in the repository. The Output Target to be used needs to be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputFields?
validfrom=24-03-01-12.34.56.123456
HTTP/1.1 400 Bad Request

{"Error": "OutputFields failed",
"Reason": "Specified timestamp '24-03-01-12.34.56.123456' has invalid format. Valid format is 'Y
YYY-MM-DD-HH.MM.SS.FFFFFF'"}
```

A timestamp with an invalid format was used in the call. The timestamp to be used must be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/OutputFields ...
HTTP/1.1 500 Internal Server Error

{"Error": "OutputFields failed",
"Reason": "Caught exception while creating result",
"Exception": "<Exception message>"}
```

If there is a problem with the internal processing of a call, this message will be returned. The placeholder <Exception message> is replaced by a message that corresponds to the problem.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/OutputFields"
HTTP/1.1 500 Internal Server Error

{"Error":"OutputFields failed","Reason":"Got SQL error message while inserting ","Object":"Outpu
tField DEFAULT.Target_MsSQL.test.dollar.nr (2024-04-11-11.39.47.955000 - 2024-05-14-05.33.24.045
999)","SQL message":"TCS0597E;70R0,WIN64_6.2.9200; Repository SQL error: TCS0256E;70R0,WIN6
4_6.2.9200; PostgreSQL libpq SQLSTATE 23505, ERROR: duplicate key value violates unique constrai
nt »outputfields_pkey« DETAIL: key »(groupid, outputtargetname, outputtablename, outputfieldseq,
outputfieldname, validfrom, validuntil)=( DEFAULT , Target_MsSQL , test.dollar , 1, nr , 2024-0
4-11-11.39.47.955000, 2024-05-14-05.33.24.045999)« already exists.; N/A."}
```

If there are problems (SQL errors) when inserting, updating or deleting objects in the repository, this message is returned. The affected object is given as well as the error message received from the database.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/OutputFields"
HTTP/1.1 400 Bad Request

{"Error":"OutputFields failed","Reason":"Updating 'OutputField' object in repository not possibl
e, because not all key values of 'Before' object have been specified.","Object":"OutputField DEF
AULT.Target_MsSQL.test.dollar.data (2024-04-11-11.39.47.955000 - )"}
```

If not all key values are provided to uniquely identify an object, this will be indicated with a message like this.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/OutputFields"
HTTP/1.1 400 Bad Request

{"Error":"OutputFields failed","Reason":"Updating 'OutputField' object in repository not possibl
e, because object does not exist in repository.","Object":"OutputField DEFAULT.Target_MsSQL.tes
t.dollar.cata (2024-04-11-11.39.47.955000 - 2024-05-14-05.33.24.045999)"}
```

If no object matching the key values passed is found in the repository, this will be indicated with a message like this.

Resource ReplicationObjects (Repository connections)

Using the resource ReplicationObjects repository definitions of table connections (ReplicationRule) and field connections (ReplicationLinkage) can be retrieved.

OPTIONS ReplicationObjects -> Method Overview

```
curl -H "@auth" -iX OPTIONS http://127.0.0.1:8080/ReplicationObjects
HTTP/1.1 200 OK

{"= HTTP method =                      ":"= Explanation =",
"GET    /ReplicationObjects        ":"Get information for replication object(s)",
"POST   /ReplicationObjects        ":"Add a new or replace a replication object",
"PUT    /ReplicationObjects        ":"Change a replication object",
"DELETE /ReplicationObjects        ":"Delete a replication object",
"= Optional queries for GET =      ":"= Explanation =",
"groupid=<GroupId>                   ":"Return repl. obj. for group <GroupId> only (default valu
e = 'DEFAULT')",
"inputsourcetype=<SourceType>      ":"Return repl. obj. for input source type <SourceType> onl
y",
"inputsourcename=<SourceName>      ":"Return repl. obj. for input table <SourceName> only",
"inputfieldname=<InputFieldName>   ":"Return repl. obj. for input field <InputFieldName> only",
"inputvalidfrom=<InputValidFrom>   ":"Return repl. obj. valid at <InputValidFrom> only",
"outputtargetname=<TargetName>     ":"Return repl. obj. for Output Target <TargetName> only",
"outputtablename=<OutputTable>     ":"Return repl. obj. for output table <OutputTable> only",
"outputfieldname=<OutputFieldName> ":"Return repl. obj. for output field <OutputFieldName> onl
y",
"outputvalidfrom=<OutputValidFrom> ":"Return repl. obj. valid at <OutputValidFrom> only",
"details=<Y|N>                     ":"Returned result: 'N' key fields only, 'Y' all fields (defa
ult value = 'N')",
"= Request data =                  ":"= Explanation =",
"Data in JSON format               ":"Use data to specify the attributes of the replication obje
ct(s) to insert/update/delete"}
```

GET ReplicationObjects -> Information about connections

The simple call without further query parameters returns all table and field connections of the 'DEFAULT' (standard) group as a result.

The result is returned in JSON notation. It informs whether the output is with all details or not, the individual table connections as JSON objects within a JSON array called ‘ReplicationRules’ and the individual field connections as JSON objects within a JSON array called ‘ReplicationLinkages’. The key names of the key-value pairs in these objects correspond to the names of the fields of the repository objects (here, for example, the field names ‚GroupId‘, ‚InputSourceType‘, ‚OutputTargetName‘, ‚InputValidFrom‘, ... of the repository tables 'ReplicationRules' and 'ReplicationLinkage').

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/ReplicationObjects
HTTP/1.1 200 OK

{"Details": false,
"ReplicationRules": [
{"GroupId": "DEFAULT",
"InputSourceType": "BIGDATA_JSON",
"InputSourceName": "DATENREPO",
"OutputTargetName": "Bigdata_Kafka",
"OutputTableName": "public.test_DatenRepo",
"InputValidFrom": "0000-00-00-00.00.00.000000",
"InputValidUntil": "9999-99-99-99.99.99.999999",
"OutputValidFrom": "2021-12-08-15.10.56.500000",
"OutputValidUntil": "9999-99-99-99.99.99.999999"},
...
{"GroupId": "DEFAULT",
"InputSourceType": "POSTGRESQL",
"InputSourceName": "postgres.public.alltypes",
"OutputTargetName": "azure_mssql",
"OutputTableName": "dbo.alltypes",
"InputValidFrom": "0000-00-00-00.00.00.000000",
"InputValidUntil": "9999-99-99-99.99.99.999999",
"OutputValidFrom": "2021-08-03-12.28.23.131000",
"OutputValidUntil": "9999-99-99-99.99.99.999999"}
],
"ReplicationLinkages": [
{"GroupId": "DEFAULT",
"InputSourceType": "ADABAS",
"InputSourceName": "00001.00001",
"OutputTargetName": "Postgres",
"OutputTableName": "T00001_00001",
"InputFieldName": "ADABAS_AA",
"OutputFieldName": "ADABAS_AA",
"InputValidFrom": "0000-00-00-00.00.00.000000",
"InputValidUntil": "9999-99-99-99.99.99.999999",
"OutputValidFrom": "2024-02-06-08.29.37.631000",
"OutputValidUntil": "9999-99-99-99.99.99.999999"
},
...
{"GroupId": "DEFAULT",
"InputSourceType": "VSAM",
"InputSourceName": "VSAM.KVBEST.FILE",
"OutputTargetName": "Bigdata_Kafka",
"OutputTableName": "axeltest_s002_schl_s002_4",
"InputFieldName": "S002_TSL",
"OutputFieldName": "s002_tsl",
"InputValidFrom": "0000-00-00-00.00.00.000000",
"InputValidUntil": "9999-99-99-99.99.99.999999",
"OutputValidFrom": "2021-11-29-12.18.15.500000",
"OutputValidUntil": "9999-99-99-99.99.99.999999"}]}
```

The call can be restricted by specifying additional parameters in the query data. Only the connections that match the criteria of the parameters are returned.

The following parameters are available:

Parameter	Description
groupid	Restricts the determined connections to a group (the default is 'DEFAULT')

Parameter	Description
inputsourcetype	Restricts the determined connections to a source type
inputsourcename	Restricts the determined connections to an input table name
inputfieldname	Restricts the determined connections to an input field name
inputvalidfrom	Restricts the determined connections to a timestamp at which the input table must be valid
outputtargetname	Restricts the determined connections to an Output Target
outputtablename	Restricts the determined connections to an output table name
outputfieldname	Restricts the determined connections to an output field name
outputvalidfrom	Restricts the determined connections to a timestamp at which the output table must be valid
details	Used to set the level of detail of the return. The possible values are: Y: all fields of the repository object N: only the key fields of the repository object (default)

Note:

- If all restrictive parameters for the input part ('groupid', ‚inputsourcetype', ‚inputsourcename', ‚inputfieldname' and ‚inputvalidfrom') or all restrictive parameters for the output part ('groupid', ‚outputtargetname', ‚outputtablename', ‚outputfieldname' and ‚outputvalidfrom') are set, 'Y' is automatically used for 'details'. In this case the result consists of only a few connections with all the information from the repository definition of the corresponding connection.
- If these parameters contain special characters such as spaces, <, >, ^, $, etc., they must be encoded accordingly before sending to the REST API server. A table with some characters and their encoding can be found in chapter „Resource Import".

Examples of using the parameters

Only the connections of source type "ADABAS" should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/ReplicationObjects?inputsourcetype=ADABAS
HTTP/1.1 200 OK

{"Details": false,
"ReplicationRules": [
{"GroupId": "DEFAULT",
"InputSourceType": "ADABAS",
"InputSourceName": "00001.00001",
"OutputTargetName": "Target_Postgres",
"OutputTableName": "T00001_00001",
"InputValidFrom": "0000-00-00-00.00.00.000000",
"InputValidUntil": "9999-99-99-99.99.99.999999",
"OutputValidFrom": "2024-02-06-08.29.37.631000",
"OutputValidUntil": "9999-99-99-99.99.99.999999"}
],
"ReplicationLinkages": [
{"GroupId": "DEFAULT",
"InputSourceType": "ADABAS",
"InputSourceName": "00001.00001",
"OutputTargetName": "Target_Postgres",
"OutputTableName": "T00001_00001",
"InputFieldName": "ADABAS_AA",
"OutputFieldName": "ADABAS_AA",
"InputValidFrom": "0000-00-00-00.00.00.000000",
"InputValidUntil": "9999-99-99-99.99.99.999999",
"OutputValidFrom": "2024-02-06-08.29.37.631000",
"OutputValidUntil": "9999-99-99-99.99.99.999999"},
...
{"GroupId": "DEFAULT",
"InputSourceType": "ADABAS",
"InputSourceName": "01111.01111",
"OutputTargetName": "Target_Oracle",
"OutputTableName": "T01111_01111",
"InputFieldName": "ISN",
"OutputFieldName": "ISN",
"InputValidFrom": "0000-00-00-00.00.00.000000",
"InputValidUntil": "9999-99-99-99.99.99.999999",
"OutputValidFrom": "2024-01-08-14.27.31.500000",
"OutputValidUntil": "9999-99-99-99.99.99.999999"}]}
```

Only the connections of output table “test.ada_employees” for Output Target “SQL Server” should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/ReplicationObjects?outputtargetname=SQL%20Server&o
utputtablename=test.employees
HTTP/1.1 200 OK

{"Details": false,
"ReplicationLinkages": [
{"GroupId": "DEFAULT",
"InputSourceType": "ADABAS",
"InputSourceName": "00094.00025",
"OutputTargetName": "SQL Server",
"OutputTableName": "test.ada_employees",
"InputFieldName": "ADA_FIRST_NAME",
"OutputFieldName": "ada_first_name",
"InputValidFrom": "0000-00-00-00.00.00.000000",
"InputValidUntil": "9999-99-99-99.99.99.999999",
"OutputValidFrom": "2020-11-15-20.48.19.500000",
"OutputValidUntil": "9999-99-99-99.99.99.999999"},
...
{"GroupId": "DEFAULT",
"InputSourceType": "ADABAS",
"InputSourceName": "00094.00025",
"OutputTargetName": "SQL Server",
"OutputTableName": "test.ada_employees",
"InputFieldName": "ISN",
"OutputFieldName": "isn",
"InputValidFrom": "0000-00-00-00.00.00.000000",
"InputValidUntil": "9999-99-99-99.99.99.999999",
"OutputValidFrom": "2020-11-15-20.48.19.500000",
"OutputValidUntil": "9999-99-99-99.99.99.999999"}]}
```

Only the connection of field “PRICE” of table “test.artikel” for Output Target “Target_Postgres” should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/ReplicationObjects?outputtargetname=Target_Postgre
s&outputtablename=test.artikel&
outputfieldname=PRICE
HTTP/1.1 200 OK

{"Details": false,
"ReplicationLinkages": [
{"GroupId": "DEFAULT",
"InputSourceType": "DB2/MVS",
"InputSourceName": "DBCG.DEMO.ARTIKEL",
"OutputTargetName": "Target_Postgres",
"OutputTableName": "test.artikel",
"InputFieldName": "PREIS",
"OutputFieldName": "PREIS",
"InputValidFrom": "2021-08-05-13.58.58.613013",
"InputValidUntil": "9999-99-99-99.99.99.999999",
"OutputValidFrom": "2022-04-27-12.20.07.306000",
"OutputValidUntil": "2023-06-13-17.31.26.499999"
},
{"GroupId": "DEFAULT",
"InputSourceType": "DB2/MVS",
"InputSourceName": "DBCG.DEMO.ARTIKEL",
"OutputTargetName": "Target_Postgres",
"OutputTableName": "test.artikel",
"InputFieldName": "PREIS",
"OutputFieldName": "PREIS",
"InputValidFrom": "2021-08-05-13.58.58.613013",
"InputValidUntil": "9999-99-99-99.99.99.999999",
"OutputValidFrom": "2023-06-13-17.31.26.500000",
"OutputValidUntil": "9999-99-99-99.99.99.999999"}]}
```

Note: The output table was reimported, which is why two versions of the table, the field and therefore also the connection exist with different validity ranges.

Only the connection of field “PRICE” of table “test.article” for Output Target “Target_Postgres”, valid at time “2023-06-13-17.31.26.500000”, should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/ReplicationObjects?outputtargetname=Target_Postgre
```

```
s&outputtablename=test.artikel&
outputfieldname=PRICE&outputvalidfrom=2023-06-13-17.31.26.500000
HTTP/1.1 200 OK

{"Details": false,
"ReplicationLinkages": [
{"GroupId": "DEFAULT",
"InputSourceType": "DB2/MVS",
"InputSourceName": "DBCG.DEMO.ARTIKEL",
"OutputTargetName": "Target_Postgres",
"OutputTableName": "test.artikel",
"InputFieldName": "PREIS",
"OutputFieldName": "PREIS",
"InputValidFrom": "2021-08-05-13.58.58.613013",
"InputValidUntil": "9999-99-99-99.99.99.999999",
"OutputValidFrom": "2023-06-13-17.31.26.500000",
"OutputValidUntil": "9999-99-99-99.99.99.999999",
"Description": "",
"ConvertCondType": 0,
"ConvertCondParm1": "",
"ConvertCondParm2": "",
"ConvertCondParm3": "",
"ConvertCondScope": "R",
"ConvertProcType": 1,
"ConvertProcParms": ""}]}
```

Note: By defining all output parameters, the parameter 'details=Y' is automatically set and the output contains all relevant information about the connection.

POST ReplicationObjects -> Inserting connections

The call allows table connections (ReplicationRule) and field connections (ReplicationLinkage) to be inserted into the repository.

For the insert all field values of the objects must be stored in a file in JSON format, and this file is then sent in the request data. The names of the field values correspond to the field names in the corresponding repository tables, and are not case sensitive.

The individual objects are grouped into JSON arrays. The identifiers “ReplicationRules” and “ReplicationLinkages” are available for these groups, and are also not case sensitive.

A table connection between an input and an output table is to be inserted into the repository table ReplicationRules. To do this, a JSON file “data.json” with the following content (shown in abbreviated form) is created.

```
{
"ReplicationRules":[
{"GroupId":"DEFAULT",
"InputSourceType":"DB2/MVS",
"InputSourceName":"DB9D.DEMO.CUSTOMERS",
"OutputTargetName":"kafka",
"OutputTableName":"customers",
"InputValidFrom":"2024-04-10-13.31.20.387494",
"InputValidUntil":"9999-99-99-99.99.99.999999",
"OutputValidFrom":"2024-04-10-13.32.16.569000",
"OutputValidUntil":"9999-99-99-99.99.99.999999",
"Description":"",
"ReplicationType":"W"
...
"BigDataActionField":"",
"BigDataActionInsert":"",
"BigDataActionUpdate":"",
"BigDataActionDelete":""
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/ReplicationObjects"
HTTP/1.1 200 OK

{"Replication Rules inserted":[
{"GroupId":"DEFAULT","InputSourceType":"DB2/MVS", "InputSourceName":"DB9D.DEMO.CUSTOMERS","Outpu
tTargetName":"kafka","OutputTableName":"customers","InputValidFrom":"2024-04-10-13.31.20.38749
4", "InputValidUntil":"9999-99-99-99.99.99.999999","OutputValidFrom":"2024-04-10-13.32.16.56900
0","OutputValidUntil":"9999-99-99-99.99.99.999999"}
]}
```

Two field connections between an input and an output table are to be inserted into the repository table ReplicationLinkage. To do this, a JSON file “data.json” with the following content is created.

```
{
"ReplicationLinkages":[
{"GroupId":"DEFAULT",
"InputSourceType":"DB2/MVS",
"InputSourceName":"DB9D.DEMO.CUSTOMERS",
"OutputTargetName":"kafka",
"OutputTableName":"customers",
"InputFieldName":"ADDRESS",
"OutputFieldName":"address",
"InputValidFrom":"2024-04-10-13.31.20.387494",
"InputValidUntil":"9999-99-99-99.99.99.999999",
"OutputValidFrom":"2024-04-10-13.32.16.569000",
"OutputValidUntil":"9999-99-99-99.99.99.999999",
"Description":"",
"ConvertCondType":0,
...
"ConvertProcParms":""
},
{"GroupId":"DEFAULT",
"InputSourceType":"DB2/MVS",
"InputSourceName":"DB9D.DEMO.CUSTOMERS",
"OutputTargetName":"kafka"
"OutputTableName":"customers",
"InputFieldName":"CITY",
"OutputFieldName":"city",
"InputValidFrom":"2024-04-10-13.31.20.387494"
"InputValidUntil":"9999-99-99-99.99.99.999999",
"OutputValidFrom":"2024-04-10-13.32.16.569000"
"OutputValidUntil":"9999-99-99-99.99.99.999999",
"Description":"",
...
"ConvertProcParms":""
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/ReplicationObjects"
HTTP/1.1 200 OK

{"Replication Linkages inserted":[
{"GroupId":"DEFAULT","InputSourceType":"DB2/MVS", "InputSourceName":"DB9D.DEMO.CUSTOMERS","Outpu
tTargetName":"kafka","OutputTableName":"customers","InputFieldName":"ADDRESS","OutputFieldNam
e":"address","InputValidFrom":"2024-04-10-13.31.20.387494","InputValidUntil":"9999-99-99-99.99.9
9.999999","OutputValidFrom":"2024-04-10-13.32.16.569000","OutputValidUntil":"9999-99-99-99.99.9
9.999999"},
{"GroupId":"DEFAULT","InputSourceType":"DB2/MVS", "InputSourceName":"DB9D.DEMO.CUSTOMERS","Outpu
tTargetName":"kafka","OutputTableName":"customers","InputFieldName":"CITY","OutputFieldName":"ci
ty","InputValidFrom":"2024-04-10-13.31.20.387494","InputValidUntil":"9999-99-99-99.99.99.99999
9","OutputValidFrom":"2024-04-10-13.32.16.569000","OutputValidUntil":"9999-99-99-99.99.99.99999
9"}
]}
```

PUT ReplicationObjects -> Updating connections

The call allows table connections (ReplicationRule) and field connections (ReplicationLinkage) to be updated in the repository.

For the update all field values of the objects must be stored in a file in JSON format, and this file is then sent in the request data. The names of the field values correspond to the field names in the corresponding repository tables, and are not case sensitive. The names of the key values of the object to be updated is preceded by the prefix “Before_”.

The individual objects are grouped into JSON arrays. The identifiers “ReplicationRules” and “ReplicationLinkages” are available for these groups, and are also not case sensitive.

A table connection between an input and an output table is to be updated in the repository table ReplicationRules. To do this, a JSON file “data.json” with the following content (shown in abbreviated form) is created.

```
{
"ReplicationRules":[
{"Before_GroupId":"DEFAULT",
"Before_InputSourceType":"DB2/MVS",
"Before_InputSourceName":"DB9D.DEMO.CUSTOMERS",
"Before_OutputTargetName":"kafka",
"Before_OutputTableName":"customers",
"Before_InputValidFrom":"2024-04-10-13.31.20.387494",
"Before_OutputValidFrom":"2024-04-10-13.32.16.569000",
"GroupId":"DEFAULT",
"InputSourceType":"DB2/MVS",
"InputSourceName":"DB9D.DEMO.CUSTOMERS",
"OutputTargetName":"kafka",
"OutputTableName":"customers",
"InputValidFrom":"2024-04-10-13.31.20.387494",
"InputValidUntil":"9999-99-99-99.99.99.999999",
"OutputValidFrom":"2024-04-10-13.32.16.569000",
"OutputValidUntil":"9999-99-99-99.99.99.999999",
"Description":"Changed to Journalreplication",
"ReplicationType":"J",
...
"BigDataActionDelete":""
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/ReplicationObjects"
HTTP/1.1 200 OK
{"Replication Rules updated":[
{"GroupId":"DEFAULT","InputSourceType":"DB2/MVS","InputSourceName":"DB9D.DEMO.CUSTOMERS","Output
TargetName":"kafka","OutputTableName":"customers","InputValidFrom":"2024-04-10-13.31.20.38749
4","OutputValidFrom":"2024-04-10-13.32.16.569000"}
]}
```

Two field connections between an input and an output table are to be inserted into the repository table ReplicationLinkage. To do this, a JSON file “data.json” with the following content (shown in abbreviated form) is created.

```
{
"ReplicationLinkages":[
{"Before_GroupId":"DEFAULT",
"Before_InputSourceType":"DB2/MVS",
"Before_InputSourceName":"DB9D.DEMO.CUSTOMERS",
"Before_OutputTargetName":"kafka",
"Before_OutputTableName":"customers",
"Before_InputFieldName":"ADDRESS",
"Before_OutputFieldName":"address",
"Before_InputValidFrom":"2024-04-10-13.31.20.387494",
"Before_OutputValidFrom":"2024-04-10-13.32.16.569000",
"GroupId":"DEFAULT",
"InputSourceType":"DB2/MVS",
"InputSourceName":"DB9D.DEMO.CUSTOMERS",
"OutputTargetName":"kafka",
"OutputTableName":"customers",
"InputFieldName":"ADDRESS",
"OutputFieldName":"address",
"InputValidFrom":"2024-04-10-13.31.20.387494",
"InputValidUntil":"9999-99-99-99.99.99.999999",
"OutputValidFrom":"2024-04-10-13.32.16.569000",
"OutputValidUntil":"9999-99-99-99.99.99.999999",
"Description":"Changed Conditionscope to F",
...
"ConvertCondScope":"F",
…
},
{"Before_GroupId":"DEFAULT",
"Before_InputSourceType":"DB2/MVS",
"Before_InputSourceName":"DB9D.DEMO.CUSTOMERS",
"Before_OutputTargetName":"kafka"
"Before_OutputTableName":"customers",
"Before_InputFieldName":"CITY",
"Before_OutputFieldName":"city",
"Before_InputValidFrom":"2024-04-10-13.31.20.387494"
"Before_OutputValidFrom":"2024-04-10-13.32.16.569000"
"GroupId":"DEFAULT",
"InputSourceType":"DB2/MVS",
"InputSourceName":"DB9D.DEMO.CUSTOMERS",
"OutputTargetName":"kafka"
"OutputTableName":"customers",
"InputFieldName":"CITY",
"OutputFieldName":"city",
"InputValidFrom":"2024-04-10-13.31.20.387494"
"InputValidUntil":"9999-99-99-99.99.99.999999",
"OutputValidFrom":"2024-04-10-13.32.16.569000"
"OutputValidUntil":"9999-99-99-99.99.99.999999",
"Description":"Changed Conditionscope to F"
...
"ConvertCondScope":"F",
...
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/ReplicationObjects"
HTTP/1.1 200 OK

{"Replication Linkages updated":[
{"GroupId":"DEFAULT","InputSourceType":"DB2/MVS","InputSourceName":"DB9D.DEMO.CUSTOMERS","Output
TargetName":"kafka","OutputTableName":"customers","InputFieldName":"ADDRESS","OutputFieldNam
e":"address","InputValidFrom":"2024-04-10-13.31.20.387494","OutputValidFrom":"2024-04-10-13.32.1
6.569000"},
{"GroupId":"DEFAULT","InputSourceType":"DB2/MVS","InputSourceName":"DB9D.DEMO.CUSTOMERS","Output
TargetName":"kafka","OutputTableName":"customers","InputFieldName":"CITY","OutputFieldName":"cit
y","InputValidFrom":"2024-04-10-13.31.20.387494","OutputValidFrom":"2024-04-10-13.32.16.569000"}
]}
```

DELETE ReplicationObjects -> Deleting connections

The call allows table connections (ReplicationRule) and field connections (ReplicationLinkage) to be deleted from the repository.

To delete, the key values of the object to be deleted must be saved in a file in JSON format, and this file is then sent in the request data. The names of the field values correspond to the field names in the corresponding repository tables, and are not case sensitive.

The individual objects are grouped into JSON arrays. The identifiers “ReplicationRules” and “ReplicationLinkages” are available for these groups, and are also not case sensitive.

A table connection between an input and an output table is to be deleted from the repository table ReplicationRules. To do this, a JSON file “data.json” with the following content is created.

```
{
"ReplicationRules":[
{"GroupId":"DEFAULT",
"InputSourceType":"DB2/MVS",
"InputSourceName":"DB9D.DEMO.CUSTOMERS",
"OutputTargetName":"kafka",
"OutputTableName":"customers",
"InputValidFrom":"2024-04-10-13.31.20.387494",
"OutputValidFrom":"2024-04-10-13.32.16.569000"
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX DELETE "http://127.0.0.1:8080/ReplicationObjects"
HTTP/1.1 200 OK

{"Replication Rules deleted":[
{"GroupId":"DEFAULT","InputSourceType":"DB2/MVS","InputSourceName":"DB9D.DEMO.CUSTOMERS","Output
TargetName":"kafka","OutputTableName":"customers","InputValidFrom":"2024-04-10-13.31.20.38749
4","OutputValidFrom":"2024-04-10-13.32.16.569000"}
]}
```

Two field connections between an input and an output table are to be deleted from the repository table ReplicationLinkage. To do this, a JSON file “data.json” with the following content is created.

```
{
"ReplicationLinkages":[
{"GroupId":"DEFAULT",
"InputSourceType":"DB2/MVS",
"InputSourceName":"DB9D.DEMO.CUSTOMERS",
"OutputTargetName":"kafka",
"OutputTableName":"customers",
"InputFieldName":"ADDRESS",
"OutputFieldName":"address",
"InputValidFrom":"2024-04-10-13.31.20.387494",
"OutputValidFrom":"2024-04-10-13.32.16.569000"
},
{"GroupId":"DEFAULT",
"InputSourceType":"DB2/MVS",
"InputSourceName":"DB9D.DEMO.CUSTOMERS",
"OutputTargetName":"kafka"
"OutputTableName":"customers",
"InputFieldName":"CITY",
"OutputFieldName":"city",
"InputValidFrom":"2024-04-10-13.31.20.387494",
"OutputValidFrom":"2024-04-10-13.32.16.569000"
} ] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX DELETE "http://127.0.0.1:8080/ReplicationObjects"
HTTP/1.1 200 OK
```

```
{"Replication Linkages deleted":[
{"GroupId":"DEFAULT","InputSourceType":"DB2/MVS", "InputSourceName":"DB9D.DEMO.CUSTOMERS","Outpu
tTargetName":"kafka","OutputTableName":"customers","InputFieldName":"ADDRESS","OutputFieldNam
e":"address","InputValidFrom":"2024-04-10-13.31.20.387494","OutputValidFrom":"2024-04-10-13.32.1
6.569000"},
{"GroupId":"DEFAULT","InputSourceType":"DB2/MVS", "InputSourceName":"DB9D.DEMO.CUSTOMERS","Outpu
tTargetName":"kafka","OutputTableName":"customers","InputFieldName":"CITY","OutputFieldName":"ci
ty","InputValidFrom":"2024-04-10-13.31.20.387494","OutputValidFrom":"2024-04-10-13.32.16.56900
0"}
]}
```

Error messages for resource ReplicationObjects

If an error occurs when executing an option of resource ReplicationObject, an HTTP return code not equal to 200 is returned.

Below is a brief overview of possible error messages and their causes:

```
curl -H "@auth" -iX GETT http://127.0.0.1:8080/ReplicationObjects
HTTP/1.1 400 Bad Request

{"Error": "Method 'GETT' is not supported"}
```

An incorrect method was used in the call. The call needs to be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/ReplicationObjects?groupid=MyGroup
HTTP/1.1 400 Bad Request

{"Error": "ReplicationObjects failed",
"Reason": "Specified groupid 'MyGroup' is not defined in repository."}
```

A group was used in the call that is not defined in the repository. The group to be used must be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/ReplicationObjects?
outputtargetname=MyTarget
HTTP/1.1 400 Bad Request

{"Error": "ReplicationObjects failed",
"Reason": "Specified Output Target name 'DEFAULT.MyTarget' is not defined in repository."}
```

An Output Target was used in the call that is not defined in the repository. The Output Target to be used needs to be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/ReplicationObjects?
inputvalidfrom=24-03-01-12.34.56.123456
HTTP/1.1 400 Bad Request

{"Error": "ReplicationObjects failed",
"Reason": "Specified input timestamp '24-03-01-12.34.56.123456' has invalid format. Valid format
is 'YYYY-MM-DD-HH.MM.SS.FFFFFF'"}
```

An input timestamp with an invalid format was used in the call. The timestamp to be used must be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/ReplicationObjects?
outputvalidfrom=24-03-01-12.34.56.123456
HTTP/1.1 400 Bad Request

{"Error": "ReplicationObjects failed",
"Reason": "Specified output timestamp '24-03-01-12.34.56.123456' has invalid format. Valid forma
t is 'YYYY-MM-DD-HH.MM.SS.FFFFFF'"}
```

An output timestamp with an invalid format was used in the call. The timestamp to be used must be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/ ReplicationObjects ...
HTTP/1.1 500 Internal Server Error

{"Error": " ReplicationObjects failed",
"Reason": "Caught exception while creating result",
"Exception": "<Exception message>"}
```

If there is a problem with the internal processing of a call, this message will be returned. The placeholder <Exception message> is replaced by a message that corresponds to the problem.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/ReplicationObjects"
HTTP/1.1 500 Internal Server Error

{"Error":"ReplicationObjects failed","Reason":"Got SQL error message while inserting ","SQL mess
age":"TCS0597E;70R0,WIN64_6.2.9200; Repository SQL error: TCS0256E;70R0,WIN64_6.2.9200; PostgreS
QL libpq SQLSTATE 23505, ERROR: duplicate key value violates unique constraint »replicationrule
s_pkey« DETAIL: key »(groupid, inputsourcetype, inputsourcename, outputtargetname, outputtablena
me, inputvalidfrom, outputvalidfrom)=(DEFAULT , DB2/MVS , DB9D.DEMO.CUSTOMERS , kafka , customer
s , 2024-04-10-13.31.20.387494, 2024-04-10-13.32.16.569000)« already exists.; N/A."}
```

If there are problems (SQL errors) when inserting, updating or deleting objects in the repository, this message is returned. The affected object is given as well as the error message received from the database.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/ReplicationObjects"
HTTP/1.1 400 Bad Request

{"Error":"ReplicationObjects failed","Reason":"Updating 'ReplicationRule' object in repository n
ot possible, because not all key values of 'Before' object have been specified.","Object":"Repli
cationRule DEFAULT.DB2/MVS.DB9D.DEMO.CUSTOMERS (2024-04-10-13.31.20.387494) - RESTAPI.kafka. (20
24-04-10-13.32.16.569000)"}
```

If not all key values are provided to uniquely identify an object, this will be indicated with a message like this.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/ReplicationObjects"
HTTP/1.1 400 Bad Request

{"Error":"ReplicationObjects failed","Reason":"Updating 'ReplicationRule' object in repository n
ot possible, because object does not exist in repository.","Object":"ReplicationRule RESTAPI.DB
2/MVS.DB9D.DEMO.CUSTOMERS (2024-04-10-13.31.20.387494) - DEFAULT.kafka.castomers (2024-04-10-1
3.32.16.569000)"}
```

If no object matching the key values passed is found in the repository, this will be indicated with a message like this.

Resource RequestManagements

Using the resource RequestManagements repository definitions of table RequestManagement can be retrieved, created, changed and deleted.

OPTIONS RequestManagements -> Method Overview

```
curl -H "@auth" -iX OPTIONS http://127.0.0.1:8080/RequestManagements
HTTP/1.1 200 OK

{"= HTTP method =              ":"= Explanation =",
"GET    /RequestManagements   ":"Get information for RequestManagement(s)",
"POST   /RequestManagements   ":"Add a new or replace a RequestManagement",
"PUT    /RequestManagements   ":"Change a RequestManagement",
"DELETE /RequestManagements   ":"Delete a RequestManagement",
"= Optional queries for GET = ":"= Explanation =",
"uniqueid=<UniqueId>          ":"Return RequestManagement with unique id <UniqueId> only",
"entrytimestamp=<TS>          ":"Return RequestManagement for entry timestamp <TS> only",
"action=<Action>              ":"Return RequestManagement for action <Action> only",
"details=<Y|N>                ":"Returned result: 'N' key fields only, 'Y' all fields (default v
alue = 'N')",
"= Request data =             ":"= Explanation =",
"Data in JSON format          ":"Use data to specify the attributes of the RequestManagement(s)
to post/put/delete"}
```

GET RequestManagements -> Information about RequestManagements

The simple call without further query parameters returns all entries of table ‘RequestManagement’.

The result is returned in JSON notation. It informs whether the output is with all details or not and the individual RequestManagement entries as JSON objects within a JSON array called ‘Results’. The key names of the key-value pairs in these objects correspond to the names of the fields of the repository objects (here, for example, the field names 'UniqueId', 'EntryTimestamp' and 'Action' of the repository table 'RequestManagement'). Although the field ‘Action’ is not part of the key of the ‘RequestManagement’ table, it is an important attribute for grouping the entries and is therefore output with the two key fields.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/RequestManagements
HTTP/1.1 200 OK

{"Details":false,
"Results":[
{"UniqueId":"287f9054-2a3d-4440-af77-e3348480299c",
"EntryTimestamp":"2024-06-20-15.24.00.000000",
"Action":"Replicate"},
{"UniqueId":"3580ae0e-1598-4ece-94a1-b079f94eef87",
"EntryTimestamp":"2024-06-25-18.22.00.000000",
"Action":"Replicate"}
] }
```

The call can be restricted by specifying additional parameters in the query data. Only the RequestManagement entries that match the criteria of the parameters are returned.

The following parameters are available:

Parameter	Description
uniqueid	Restricts the determined entries to a specific id
entrytimestamp	Restricts the determined entries to a specific timestamp
action	Restricts the determined entries to a specific action (e.g. ‘Replicate’, …)
details	Used to set the level of detail of the return. The possible values are: Y: all fields of the repository object N: only the key fields (and ‘action’) of the repository object (default)

Note:

- If all restrictive parameters ('uniqueid', 'entrytimestamp' and 'action') are set, 'Y' is automatically used for 'details'. In this case the result only consists of one entry with all the information from the repository definition of this entry.
- If these parameters contain special characters such as spaces, <, >, ^, $, etc., they must be encoded accordingly before sending to the REST API server. A table with some characters and their encoding can be found in chapter „Resource Import“.

Examples of using the parameters

Only the repository entries with action ‘Replicate’ should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/RequestManagements?action=Replicate
HTTP/1.1 200 OK

{"Details":false,
"Results":[
{"UniqueId":"287f9054-2a3d-4440-af77-e3348480299c",
"EntryTimestamp":"2024-06-20-15.24.00.000000",
"Action":"Replicate"},
{"UniqueId":"3580ae0e-1598-4ece-94a1-b079f94eef87",
"EntryTimestamp":"2024-06-25-18.22.00.000000",
"Action":"Replicate"}
]}
```

Only the repository entries with action ‘Replicate’ and id ‘3580ae0e-1598-4ece-94a1-b079f94eef87’ should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/RequestManagements?action=Replicate&uniqueid=3580a
e0e-1598-4ece-94a1-b079f94eef87
HTTP/1.1 200 OK

{"Details":false,
"Results":[
{"UniqueId":"3580ae0e-1598-4ece-94a1-b079f94eef87",
"EntryTimestamp":"2024-06-25-18.22.00.000000",
"Action":"Replicate"}
]}
```

Only the repository entries with action ‘Replicate’, id ‘3580ae0e-1598-4ece-94a1-b079f94eef87’ and timestamp ‘2024-06-20-15.24.00.000000’ should be returned.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/RequestManagements?action=Replicate&uniqueid=287f9
054-2a3d-4440-af77-e3348480299c&entrytimestamp=  2024-06-20-15.24.00.000000
HTTP/1.1 200 OK

{"Details":true,
"Results":[
{"UniqueId":"287f9054-2a3d-4440-af77-e3348480299c",
"EntryTimestamp":"2024-06-20-15.24.00.000000",
"RequestorConnection":"",
"Action":"Replicate",
"Status":"Requested",
"RequestData":"{\"id\": \"287f9054-2a3d-4440-af77-e3348480299c\", \"timestamp\": \"2024-06-20T1
5:22:50+00:00\", \"source\": { \"location\": \"file:\/\/XYZ.DOMAIN.COM\/TS1234.ABC.KSDS\", \"met
adata\": { \"type\": \"VSAM\", \"ccsid\":\"1200\", \"fileType\": \"KSDS\", \"recordFormat\": \"F
ixed\", \"recordName\":\"SEGMENTNAME\", \"keyLength\": 10, \"keyPosition\": 0 }, \"schema\":{
\"type\":\"COBOL\", \"location\": \"file:\/\/XYZ.DOMAIN.COM\/TS1234.TEST.COPYLIB(ABC)\" }},\"act
ion\": \"Replicate\", \"description\": \"Replicate and bulk to PostgreSQL\"}","ResponseTimestam
p":"","ResponseData":""}
]}
```

Note: By defining all other parameters, the parameter 'details=Y' is automatically set and the output contains all relevant information about the entry.

POST RequestManagements -> Inserting RequestManagement entries

The call allows RequestManagement entries to be inserted into the repository.

For the insert all values of a RequestManagement entry must be stored in a file in JSON format, and this file is then sent in the request data. The names of the values correspond to the field names in repository table “RequestManagement”, and are not case sensitive.

The individual RequestManagement entries are embedded within the JSON array “RequestManagements”, this identifier is also not case sensitive.

An entry is to be inserted into the repository table RequestManagements. To do this, a JSON file “data.json” with the following content is created.

```
{
"RequestManagements":[
{"UniqueId":"12345678-abcd-1234-cdef-a1234567890b",
"EntryTimestamp":"2024-07-04-12.34.56.000000",
"RequestorConnection":"",
"Action":"Replicate",
"Status":"Requested",
"RequestData":"{\"id\": \"12345678-abcd-1234-cdef-a1234567890b\", \"timestamp\": \"2024-07-04T1
2:34:56+00:00\", \"source\": { \"location\": \"file:\/\/XYZ.DOMAIN.COM\/TS1234.ABC.KSDS\", \"met
adata\": { \"type\": \"VSAM\", \"ccsid\":\"1200\", \"organization\": \"KSDS\", \"recordFormat\":
\"Fixed\", \"keyLength\": 15, \"keyPosition\": 5 }, \"schema\":{ \"type\": \"Cobol\", \"locatio
n\": \"file:\/\/XYZ.DOMAIN.COM\/TS1234.TEST.COPYLIB(ABC)\" }},\"action\": \"Replicate\", \"descr
iption\": \"Replicate and bulk\"}",
"ResponseTimestamp":"",
"ResponseData":""}
]
}
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/RequestManagements"
HTTP/1.1 200 OK

{"RequestManagement inserted":[
{"UniqueId":"12345678-abcd-1234-cdef-a1234567890b",
"EntryTimestamp":"2024-07-04-12.34.56.000000"}
]}
```

PUT RequestManagements -> Updating RequestManagement entries

Calling this method allows to update RequestManagement entries in the repository.

To update, the key values of the entry to be updated as well as all values of the resulting entry must be saved in a file in JSON format, and this file is then sent in the request data. The names of the values correspond to the field names in repository table "RequestManagement", and are not case sensitive. The names of the key values of the entery to be updated is preceded by the prefix "Before_".

The individual RequestManagement entries are embedded within the JSON array "RequestManagement s", this identifier is also not case sensitive.

An entry is to be updated in the repository table RequestManagement. To do this, a JSON file "data.json" with the following content is created.

```
{
"RequestManagements":[
{"Before_UniqueId":"12345678-abcd-1234-cdef-a1234567890b",
"Before_EntryTimestamp":"2024-07-04-12.34.56.000000",
"UniqueId":"12345678-abcd-1234-cdef-a1234567890b",
"EntryTimestamp":"2024-07-04-12.34.56.000000",
"Status":"Done",
"ResponseData":"{\"id\":\"57fe7f09-a454-4532-bb1c-5cae9bdb35d2\",\"timestamp\":\"2024-07-05T13:5
5:27+00:00\",\"requestID\":\"12345678-abcd-1234-cdef-a1234567890b\",\"status\":\"Done\",\"finish
ed\":true,\"description\":\"Done\",\"impacts\":[{\"type\":\"InputTable\",\"id\":\"DEFAULT.VSAM.T
S1234.ABC.KSDS\"},{\"type\":\"OutputTable\",\"id\":\"DEFAULT.Target_Postgres.ABC\"}]}"}
]
}
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/RequestManagements"
HTTP/1.1 200 OK

{"RequestManagement updated":[
{"UniqueId":"12345678-abcd-1234-cdef-a1234567890b",
"EntryTimestamp":"2024-07-04-12.34.56.000000"}
]}
```

DELETE RequestManagements -> Deleting RequestManagement entries

Calling this method allows to delete RequestManagement entries from the repository.

To delete, the key values of the entry to be deleted must be saved in a file in JSON format, and this file is then sent in the request data. The names of the values correspond to the field names in repository table “RequestManagement”, and are not case sensitive.

The individual RequestManagement entries are embedded within the JSON array “RequestManagements”, this identifier is also not case sensitive.

An entry is to be deleted from the repository table RequestManagement. To do this, a JSON file “data.json” with the following content is created.

```
{
"RequestManagements":[
{"UniqueId":"12345678-abcd-1234-cdef-a1234567890b",
"EntryTimestamp":"2024-07-04-12.34.56.000000"
}
] }
```

This file is then passed in the call.

```
curl -d @data.json -H "@auth" -iX DELETE "http://127.0.0.1:8080/RequestManagements"
HTTP/1.1 200 OK

{"RequestManagement deleted":[
{"UniqueId":"12345678-abcd-1234-cdef-a1234567890b",
"EntryTimestamp":"2024-07-04-12.34.56.000000"}
] }
```

Error messages for resource RequestManagements

If an error occurs when executing an option of resource RequestManagements, an HTTP return code not equal to 200 is returned.

Below is a brief overview of possible error messages and their causes:

```
curl -H "@auth" -iX GETT http://127.0.0.1:8080/RequestManagements
HTTP/1.1 400 Bad Request

{"Error": "Method 'GETT' is not supported"}
```

An incorrect method was used in the call. The call needs to be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/RequestManagements?
entrytimestamp=24-06-20-15.24.00.000000
HTTP/1.1 400 Bad Request

{"Error":"RequestManagements failed",
"Reason":"Specified timestamp '24-06-20-15.24.00.000000' has invalid format. Valid format is 'YY
YY-MM-DD-HH.MM.SS.FFFFFF'"}
```

A timestamp with an invalid format was used in the call. The timestamp to be used must be corrected.

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/RequestManagements ...
HTTP/1.1 500 Internal Server Error

{"Error": " RequestManagements failed",
"Reason": "Caught exception while creating result",
"Exception": "<Exception message>"}
```

If there is a problem with the internal processing of a call, this message will be returned. The placeholder <Exception message> is replaced by a message that corresponds to the problem.

```
curl -d @data.json -H "@auth" -iX POST "http://127.0.0.1:8080/RequestManagements"
HTTP/1.1 500 Internal Server Error

{"Error":"RequestManagements failed",
"Reason":"Got SQL error message while inserting ",
"Object":"RequestManagement 12345678-abcd-1234-cdef-a1234567890b (2024-07-04-12.34.56.000000)",
"SQL message":"TCS0597E;70R1940,WIN64_6.2.9200; Repository SQL error: TCS0256E;70R1940,WIN6
4_6.2.9200; PostgreSQL libpq SQLSTATE 23505, ERROR: duplicate key value violates unique constrai
nt »requestmanagement_pkey1« DETAIL: key »(uniqueid, entrytimestamp)=(12345678-abcd-1234-cdef-a1
234567890b , 2024-07-04-12.34.56.000000)« already exists.; N\/A."}
```

If there are problems (SQL errors) when inserting, updating or deleting objects in the repository, this message is returned. The affected object is given as well as the error message received from the database.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/RequestManagements"
HTTP/1.1 400 Bad Request

{"Error":"RequestManagements failed",
"Reason":"Updating 'RequestManagement' object in repository not possible, because not all key va
lues of 'Before' object have been specified.",
"Object":"RequestManagement 12345678-abcd-1234-cdef-a1234567890b (null)"}
```

If not all key values are provided to uniquely identify an object, this will be indicated with a message like this.

```
curl -d @data.json -H "@auth" -iX PUT "http://127.0.0.1:8080/RequestManagements"
HTTP/1.1 400 Bad Request

{"Error":"RequestManagements failed",
"Reason":"Updating 'RequestManagement' object in repository not possible, because object does no
t exist in repository.",
"Object":"RequestManagement 12345678-abcd-1234-cdef-a1234567890b (2024-07-04-12.34.56.000000)"}
```

If no object matching the key values passed is found in the repository, this will be indicated with a message like this.

Resource Import

In contrast to the previous resources, which can be used to read, change or delete the contents of the repository, this resource is somewhat more complex. It is used to determine metadata information from tables from a wide variety of Data Sources (databases, Cobol copybooks, ...) and store it in the metadata repository.

This requires information that makes calling up the resource methods more extensive. In principle, importing using the REST API requires the same parameters that importing using the batch utility 'BatchImport' requires. The possible parameters and their meaning can therefore be found in the manual tcV7Batch_Components_en.pdf.

Note: However, the `-connect` parameter (and in the case of an SSL connection also the `-keystore` parameter) can be ignored. Normally the BatchImport requires these in order to establish a connection to the Agent holding the repository. Here automatically the Agent is used that is defined for starting the REST API server.

These parameters can contain special characters such as spaces, <, >, ^, $, etc. These must be encoded accordingly before sending to the REST API server.

The following tables show the most important characters and their encoding.

Character	Encoding
Space	%20
!	%21
"	%22
#	%23
$	%24
%	%25
&	%26
'	%27
(	%28
)	%29
*	%2A
+	%2B
,	%2C
-	%2D
.	%2E
/	%2F
:	%3A
;	%3B

Character	Encoding
<	%3C
=	%3D
>	%3E
?	%3F
@	%40
[	%5B
\	%5C
]	%5D
^	%5E
_	%5F
{	%7B
|	%7C
}	%7D

START Import -> Running an import

An import process is performed using the START option. To do this, the parameters for the import are passed in the query data. This is sufficient for all imports that use a database as the source of the import (Db2, Oracle, MySql, ...). For all imports that use a file as source (Cobol Copybook, Db2 Report, ...) this file is passed in the request data (--data-binary @Import_filename).

The output is in JSON notation. It provides the key values of the input and output tables created and terminated by the import to identify the objects in the metadata repository. Each of these tables is a JSON object within a JSON array named “Created InputTables”, “Created OutputTables”, “Terminated InputTables” and “Terminated OutputTables” respectively. The key names of the key-value pairs in these JSON objects correspond to the names of the repository fields of the repository objects (here, for example, the field names 'GroupId', 'SourceType', 'OutputTargetName', 'ValidFrom', ... of the repository tables 'InputTables' and 'OutputTables').

Example of importing a Db2 table:

For group “Test”, the table “DEMO.ARTIKEL" should be imported from the Data Source “DB2 zOS” (defined in the repository), and output objects and replication objects should also be created. The output objects are to be created for the Output Target “Postgres” (defined in the repository) using the output name “test.<TABLE>” (<TABLE> is a placeholder, the table name should take the name of the input object, i.e. ARTIKEL).

These requirements result in the following import parameters:

```
groupid=Test
objects=IOR
datasource=DB2 zOS
rdbmstcreator=^DEMO$
rdbmstname=^ARTIKEL$
outputtarget=Postgres
```

```
outputname=test.<TABLE>
```

As already mentioned above, the special characters used must be encoded accordingly, which results in the following query string:

```
groupid=Test&objects=IOR&datasource=DB2%20zOS&rdbmstcreator=%5EDEMO%24& rdbmstname=%5EARTIKEL%2
4&outputtarget=Postgres&outputname=test.%3CTABLE%3E
```

```
curl -H "@auth" -iX START "http://127.0.0.1:8080/Import?groupid=Test&objects=IOR&datasource=DB
2%20zOS&rdbmstcreator=%5EDEMO%24& rdbmstname=%5EARTIKEL%24&outputtarget=Postgres&outputname=tes
t.%3CTABLE%3E"
HTTP/1.1 200 OK

{"Created InputTables": [
{"GroupId": "Test",
"SourceType": "DB2/MVS",
"SourceName": "DALLASC.DEMO.ARTIKEL",
"ValidFrom": "2023-05-05-10.35.49.234140",
"ValidUntil": "9999-99-99-99.99.99.999999"}
],
"Created OutputTables": [
{"GroupId": "Test",
"OutputTargetName": "Postgres",
"OutputTable": "test.ARTIKEL",
"ValidFrom": "2023-12-21-14.41.05.112000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

An input and an output table were created.

Alternative call:

Since in this example, unlike the next example, no file is used as import source, the parameters can alternatively be sent in the request data instead of in the query data. To do this, the parameters must be stored in a file in JSON format, and this file is then sent in the request data. The advantage of this method is that the call is clearer and the above-mentioned encoding of the special characters in the JSON file can be omitted.

The content of the JSON file, e.g. “ImportParms.json”, would be as follows in the example above:

```
{"groupid": "Test",
"objects": "IOR",
"datasource": "DB2 zOS",
"rdbmstcreator": "^DEMO$",
"rdbmstname": "^ARTIKEL$",
"outputtarget": "Postgres",
"outputname": "test.<TABLE>"}
```

The import itself is then reduced to the following call:

```
curl -d @ImportParms.json -H "@auth" -iX START "http://127.0.0.1:8080/Import"
```

Example of importing a Cobol copybook:

For group “Test”, the segment “REDEF-UND-OCCURS” of the Cobol copybook “copybook.txt” should be imported, and output objects and replication objects should also be created. The output objects are to be created for the Output Target “Postgres” (defined in the repository) using the output name “test.” (is a placeholder, the table name should take the name of the segment, i.e. “REDEF_AND_OCCURS”, since '- ' will be replaced with '_' in table names). The fields determined from the copybook must be bound to a Data Source. This is done here using the VSAM file “VSAM.DEMO.ARTIKEL”, which is accessed by the Agent “zOS-Agent”. Subtables are to be created (for the occurrences in the copybook) and these subtables receive an index field, redefinitions are used as a flat field list. The content of the Cobol Copybook file is transferred in the request data of the call, which is why the key word “REQUESTDATA” is used for the file name. The file content itself is transferred using “--data-binary @copybook.txt” in the call. A temporary file with the transferred content is created in the “User Home” directory, so write and read rights are required for this directory.

These requirements result in the following import parameters:

```
groupid=Test
objects=IOR
filename=REQUESTDATA
segment=REDEF-UND-OCCURS
createsubtables=Y
createindexfield=Y
resolveredef=F
outputtarget=Postgres
outputname=test.<SEGMENT>
bindagent=zOS-Agent
bindzosvsam=VSAM.DEMO.ARTIKEL.FILE
```

Again, the special characters used must be encoded accordingly, resulting in the following query string:

```
groupid=Test&objects=IOR&filename=REQUESTDATA&segment=REDEF-UND-OCCURS& createsubtables=Y&create
indexfield=Y&resolveredef=F&outputtarget=Postgres&outputname=public.%3CSEGMENT%3E&bindagent=zOS-
Agent&bindzosvsam=VSAM.DEMO.ARTIKEL.FILE
```

```
curl --data-binary @copybook.txt -H "@auth" -iX START "http://127.0.0.1:8080/Import?groupid=Tes
t&objects=IOR&filename=REQUESTDATA&segment=REDEF-UND-OCCURS& createsubtables=Y&createindexfiel
d=Y&resolveredef=F&outputtarget=Postgres& outputname=test.%3CSEGMENT%3E&bindagent=zOS-Agent& bin
dzosvsam=VSAM.DEMO.ARTIKEL.FILE"
HTTP/1.1 200 OK

{"Created InputTables": [
{"GroupId": "Test",
"SourceType": "VSAM",
"SourceName": "VSAM.DEMO.ARTIKEL.FILE",
"ValidFrom": "0000-00-00-00.00.00.000000",
"ValidUntil": "9999-99-99-99.99.99.999999"}
],
"Created OutputTables": [
{"GroupId": "Test",
"OutputTargetName": "Postgres",
"OutputTable": "public.REDEF_UND_OCCURS",
"ValidFrom": "2024-02-28-13.51.24.444000",
"ValidUntil": "9999-99-99-99.99.99.999999"
},
{"GroupId": "Test",
"OutputTargetName": "Postgres",
"OutputTable": "public.REDEF_UND_OCCURS_KAT",
"ValidFrom": "2024-02-28-13.51.24.444000",
"ValidUntil": "9999-99-99-99.99.99.999999"
},
{"GroupId": "Test",
"OutputTargetName": "Postgres",
"OutputTable": "public.REDEF_UND_OCCURS_PLZ",
"ValidFrom": "2024-02-28-13.51.24.444000",
"ValidUntil": "9999-99-99-99.99.99.999999"}]}
```

An input table and a main output table were created, as well as two additional output tables (subtables for the OCCURS fields KAT and PLZ).

Note: An alternative call like in the Db2 example cannot take place here because the request data is used to transfer the source file to be imported (Cobol Copybook). However, the file can be transferred to the REST API server beforehand with a separate call (see chapter SENDFILE Import -> Sending a file), so that the alternative call would then be possible again.

SENDFILE Import -> Sending a file

Alternatively, a file can be sent to the REST API server before the import starts. This makes sense, for example, if a Cobol copybook contains several segments that all need to be imported. The file can then be sent once, several imports can be performed on the different segments and the file can then be deleted again.

Sending a file is done using the SENDFILE option. To do this, the name of the file is transferred in the query data and the file itself is transferred in the request data (--data-binary @Send_filename). The file is created in the “User Home” directory with the transferred content, so write and read rights are required for this directory. The same file name is then used for the actual import process as here for sending.

Example of sending a file:

The Cobol copybook “copybook.txt” should be sent to the REST API server and saved there using the name “ImportData.txt”. The file content itself is transferred using “--data-binary @copybook.txt” in the call.

For this request, only the file name (without path) is required as a parameter to specify the name of the file to be stored on the REST API server.

`filename=ImportData.txt`

```
curl --data-binary @copybook.txt -iX SENDFILE "http://127.0.0.1:8080/Import?filename=ImportDat
a.txt"
HTTP/1.1 200 OK

{"Success": "Writing data into file 'C:\Users\user\ImportData.txt' finished successfully."}
```

The file was created using the given name in the “User Home” directory.

The actual import can now access this file. For the import example from above, only the parameter filename=REQUESTDATA

would be required to be change to

filename=ImportData.txt

and the transfer of the file ("--data-binary @copybook.txt") can be omitted, which then leads to the following call:

```
curl -H "@auth" -iX START "http://127.0.0.1:8080/Import?groupid=Test&objects=IOR&filename=Import
Data.txt&segment=REDEF-UND-OCCURS& createsubtables=Y&createindexfield=Y&resolveredef=F&outputtar
get=Postgres& outputname=public.%3CSEGMENT%3E&bindagent=zOS-Agent& bindzosvsam=VSAM.DEMO.ARTIKE
L.FILE"
```

DELETEFILE Import -> Deleting a file

A file created using SENDFILE can and should be removed from the REST API server after the import has been completed.

A file is deleted using the DELETEFILE option. For this purpose, the file name of the file is passed in the query data. The file is searched for in the "User Home" directory and removed there, so write permissions are required for this directory.

Example of deleting a file:

The file "ImportData.txt" created on the REST API server using SENDFILE should be deleted again.

This request only requires the file name (without path) of the file to be deleted as a parameter.

filename=ImportData.txt

```
curl -iX DELETEFILE "http://127.0.0.1:8080/Import?filename=ImportData.txt"
HTTP/1.1 200 OK

{"Success": "Deleting file 'C:\Users\user\ImportData.txt' finished successfully."}
```

The file was deleted from the "User Home" directory.

Error messages for resource Import

If an error occurs when executing an option of resource Import, an HTTP return code not equal to 200 is returned.

In addition, there is output in the JSON data, which can contain up to three pieces of information.

"Error" always appears in the output with the problem description, e.g.

```
HTTP/1.1 500 Internal Server Error

{"Error": "Unable to retrieve system property 'user.home'"}
```

In addition, the "Reason" for the problem can be displayed for some error messages, e.g.

```
HTTP/1.1 400 Bad Request

{"Error": "Import failed",
"Reason": "No parameters specified to define the import process"}
```

Finally, the actual “Exception” is displayed if such an exception was recognized in the “Reason” as the cause of a problem, e.g.

```
HTTP/1.1 500 Internal Server Error

{"Error": "Import failed",
"Reason": "Caught exception while performing BatchImport",
"Exception": "The specified bind Agent 'zOS21-Agent' does not exist.; "}
```

Resource Export

An output table in the repository usually only represents structural information without a table in a database system. This resource is used to create the output table that matches the repository object in the database.

As with the import, information is also required for the export, which makes calling the resource methods more extensive. In principle, exporting using the REST API requires the same parameters that exporting using the batch utility 'BatchExport' requires. The possible parameters and their meaning can therefore be found in the manual tcV7Batch_Components_en.pdf.

Note: However, the -connect parameter (and in the case of an SSL connection also the -keystore parameter) can be ignored. Normally the BatchExport requires these in order to establish a connection to the Agent holding the repository. Here automatically the Agent is used that is defined for starting the REST API server.

The parameters -maxlength, -bom and -eolc are also not required when creating the export information and will therefore be ignored.

CREATE Export -> Creating the export statements

Using the CREATE option, an export process is performed that generates SQL commands. These commands can then be used to create the tables in the Output Target. For this purpose, the parameters for the export are passed in the query data.

The output is in JSON notation. It provides information about the Output Targets affected by the export and the SQL commands.

Each of these targets is returned as a JSON object within a JSON array called "OutputTargets" and contains the key values of the target to identify the object in the metadata repository and the SQL commands themselves within a JSON array called "SQL commands".

If there are no SQL commands for an Output Target (e.g. for the “BigData” type), then the output contains a corresponding “Warning” entry and the JSON array for the SQL commands is omitted.

Example of exporting all output tables connected with an input table:

For group “Test”, all output tables connected to the input table “VSAM.DEMO.ARTIKEL.FILE” (defined in the repository) should be exported. The SQL commands should end with a semicolon.

These requirements result in the following export parameters:

```
groupid=Test
inputtable= VSAM.^DEMO.ARTIKEL.FILE$
eocc=;
```

As already mentioned in the import resource, the special characters used must be encoded accordingly, resulting in the following query string:

```
groupid=Test&eocc=%3B&inputtable=VSAM.%5EVSAM.DEMO.ARTIKEL.FILE%24
```

```
curl -H "@auth" -iX CREATE "http://127.0.0.1:8080/Export?groupid=Test&
eocc=%3B&inputtable=VSAM.%5EVSAM.DEMO.ARTIKEL.FILE%24"
HTTP/1.1 200 OK

{"OutputTargets": [
{"GroupId": "Test",
"OutputTargetName": "Bigdata",
"ValidFrom": "0000-00-00-00.00.00.000000",
"Warning": "Export not necessary for output table 'public.REDEF_UND_OCCURS' because of its targe
t type 'BigData'."
},
{"GroupId": "Test",
"OutputTargetName": "Oracle",
"ValidFrom": "0000-00-00-00.00.00.000000",
"SQL commands": [
"DROP TABLE TEST.REDEF_UND_OCCURS_PLZ;",
"DROP TABLE TEST.REDEF_UND_OCCURS_KAT;",
"DROP TABLE TEST.REDEF_UND_OCCURS;",
"CREATE TABLE TEST.REDEF_UND_OCCURS ( NR DECIMAL (11) NOT NULL, TYP CHAR (1 CHAR) NOT NULL, NAME
CHAR (40 CHAR) NOT NULL, PREIS DECIMAL (10,2) NOT NULL, LIEFER DECIMAL (11) NOT NULL, EIN_HEIT C
HAR (25 CHAR) NOT NULL, BESTAND DECIMAL (6) NOT NULL, LIEFERNR DECIMAL (11) NOT NULL, FIRMA CHAR
(40 CHAR) NOT NULL, KONTAKT CHAR (30 CHAR) NOT NULL, POSITION CHAR (30 CHAR) NOT NULL, STRASSE C
HAR (60 CHAR) NOT NULL, ORT CHAR (15 CHAR) NOT NULL, REGION CHAR (15 CHAR) NOT NULL, LAND CHAR
(15 CHAR) NOT NULL, TELEFON CHAR (24 CHAR) NOT NULL, TELEFAX CHAR (24 CHAR) NOT NULL);",
"CREATE TABLE TEST.REDEF_UND_OCCURS_KAT ( NR DECIMAL (11) NOT NULL, KAT_TAB_IND NUMBER(5) NOT NU
LL, KAT DECIMAL (11) NOT NULL);",
"CREATE TABLE TEST.REDEF_UND_OCCURS_PLZ ( NR DECIMAL (11) NOT NULL, PLZ_TAB_IND NUMBER(5) NOT NU
LL, PLZ CHAR (10 CHAR) NOT NULL);",
"ALTER TABLE TEST.REDEF_UND_OCCURS_PLZ ADD CONSTRAINT pk_TEST_REDEF_UND_OCCURS_PLZ PRIMARY KEY (
NR,PLZ_TAB_IND );",
"ALTER TABLE TEST.REDEF_UND_OCCURS_KAT ADD CONSTRAINT pk_TEST_REDEF_UND_OCCURS_KAT PRIMARY KEY (
NR,KAT_TAB_IND );",
"ALTER TABLE TEST.REDEF_UND_OCCURS ADD CONSTRAINT pk_TEST_REDEF_UND_OCCURS PRIMARY KEY ( NR );",
"ALTER TABLE TEST.REDEF_UND_OCCURS_PLZ ADD CONSTRAINT fk__TEST_REDEF_UND_OCCURS_PLZ FOREIGN KEY
( NR ) REFERENCES TEST.REDEF_UND_OCCURS ( NR );",
"ALTER TABLE TEST.REDEF_UND_OCCURS_KAT ADD CONSTRAINT fk__TEST_REDEF_UND_OCCURS_KAT FOREIGN KEY
( NR ) REFERENCES TEST.REDEF_UND_OCCURS ( NR );"]
},
{"GroupId": "Test",
"OutputTargetName": "Postgres",
"ValidFrom": "0000-00-00-00.00.00.000000",
"SQL commands": [
"DROP TABLE IF EXISTS \"public\".\"REDEF_UND_OCCURS_PLZ\";",
"DROP TABLE IF EXISTS \"public\".\"REDEF_UND_OCCURS_KAT\";",
"DROP TABLE IF EXISTS \"public\".\"REDEF_UND_OCCURS\";",
"CREATE TABLE \"public\".\"REDEF_UND_OCCURS\" ( \"NR\" decimal (11) NOT NULL, \"TYP\" char (1) N
OT NULL, \"NAME\" char (40) NOT NULL, \"PREIS\" decimal (10,2) NOT NULL, \"LIEFER\" decimal (11)
NOT NULL, \"EIN_HEIT\" char (25) NOT NULL, \"BESTAND\" decimal (6) NOT NULL, \"LIEFERNR\" decima
l (11) NOT NULL, \"FIRMA\" char (40) NOT NULL, \"KONTAKT\" char (30) NOT NULL, \"POSITION\" char
(30) NOT NULL, \"STRASSE\" char (60) NOT NULL, \"ORT\" char (15) NOT NULL, \"REGION\" char (15)
NOT NULL, \"LAND\" char (15) NOT NULL, \"TELEFON\" char (24) NOT NULL, \"TELEFAX\" char (24) NOT
NULL);",
"CREATE TABLE \"public\".\"REDEF_UND_OCCURS_KAT\" ( \"NR\" decimal (11) NOT NULL, \"KAT_TAB_IN
D\" smallint NOT NULL, \"KAT\" decimal (11) NOT NULL);",
"CREATE TABLE \"public\".\"REDEF_UND_OCCURS_PLZ\" ( \"NR\" decimal (11) NOT NULL, \"PLZ_TAB_IN
D\" smallint NOT NULL, \"PLZ\" char (10) NOT NULL);",
"ALTER TABLE \"public\".\"REDEF_UND_OCCURS_PLZ\" ADD CONSTRAINT \"public_REDEF_UND_OCCURS_PLZ_p
k\"  PRIMARY KEY ( \"NR\",\"PLZ_TAB_IND\" );",
"ALTER TABLE \"public\".\"REDEF_UND_OCCURS_KAT\" ADD CONSTRAINT \"public_REDEF_UND_OCCURS_KAT_p
k\"  PRIMARY KEY ( \"NR\",\"KAT_TAB_IND\" );",
"ALTER TABLE \"public\".\"REDEF_UND_OCCURS\" ADD CONSTRAINT \"public_REDEF_UND_OCCURS_pk\"  PRIM
ARY KEY ( \"NR\" );",
"ALTER TABLE \"public\".\"REDEF_UND_OCCURS_PLZ\" ADD CONSTRAINT \"public_REDEF_UND_OCCURS_PLZ_f
k\"   FOREIGN KEY ( \"NR\" )  REFERENCES \"public\".\"REDEF_UND_OCCURS\" ( \"NR\" ) ON UPDATE CA
SCADE;",
"ALTER TABLE \"public\".\"REDEF_UND_OCCURS_KAT\" ADD CONSTRAINT \"public_REDEF_UND_OCCURS_KAT_f
k\"   FOREIGN KEY ( \"NR\" )  REFERENCES \"public\".\"REDEF_UND_OCCURS\" ( \"NR\" ) ON UPDATE CA
SCADE;"]}]}
```

The input table is connected to three output tables located in different Output Targets.

A warning is issued for the first target (“Bigdata”) because there are no DDL statements for this target type, and therefore the JSON array for the SQL commands is omitted.

For the other two targets (“Oracle” and “Postgres”), the key values of the Output Targets in the repository are returned, as well as the SQL commands themselves for both.

Alternative call:

As with the resource import, the parameters can also be sent in the request data instead of in the query data. To do this, the parameters must be stored in a file in JSON format, and this file is then sent in the request data. The advantage of this method is that the call is clearer and the above-mentioned encoding of the special characters in the JSON file can be omitted.

The content of the JSON file, e.g. “ExportParms.json”, would be as follows in the example above:

```
{"groupid": "Test",
"inputtable": "VSAM.^VSAM.DEMO.ARTIKEL.FILE$",
"eocc": ";"}
```

The export itself is then reduced to the following call:

```
curl -d @ExportParms.json -H "@auth" -iX CREATE http://127.0.0.1:8080/Export
```

RUN Export -> Executing the export statements

Using the RUN option, an export process is performed that executes the SQL commands passed in the request data to create the tables in the Output Target.

This request data in JSON format has the same format as the output of the CREATE export option, i.e. information about the respective Output Target and the SQL commands themselves.

```
{"OutputTargets": [
{"GroupId": "mygroup",
"OutputTargetName": "targetname_1",
"ValidFrom": "0000-00-00-00.00.00.000000"
},
{"GroupId": "mygroup",
"OutputTargetName": "targetname_2",
"ValidFrom": "0000-00-00-00.00.00.000000",
"SQL commands": [
"Command 1",
...
"Command x"]}
...
{"GroupId": "mygroup",
"OutputTargetName": "targetname_n",
"ValidFrom": "0000-00-00-00.00.00.000000",
"SQL commands": [
"Command 1",
...
"Command y"]}]}
```

The input is returned in the output again, with each command (now key of a JSON object entry) supplemented with information (value of the JSON object entry) that describes the result of the execution (Success, Warning or Error). In case of a warning or error, the message from the database is also added. If an error occurs, processing is aborted and no further SQL commands are executed.

For example, a warning can occur if a DROP is applied to a table that does not exist. Such an "error" is negligible and is therefore returned as a warning.

Example of executing SQL commands using export:

After the SQL commands have been created using the CREATE option and have been given to the user for viewing, these (possibly revised) commands can now serve as input for executing the commands. For this purpose, they are saved in a file (e.g. "Export_Data.json") and transmitted in the request data.

No query data is required.

```
curl -d @Export_Data.json -H "@auth" -iX RUN http://127.0.0.1:8080/Export
HTTP/1.1 200 OK

{"OutputTargets": [
{"GroupId": "Test",
"OutputTargetName": "Bigdata",
"ValidFrom": "0000-00-00-00.00.00.000000"
},
{"GroupId": "Test",
"OutputTargetName": "Oracle",
"ValidFrom": "0000-00-00-00.00.00.000000",
"SQL commands": [
{"DROP TABLE TEST.REDEF_UND_OCCURS_PLZ": "Success"
},
...
{"ALTER TABLE TEST.REDEF_UND_OCCURS_KAT ADD CONSTRAINT fk__TEST_REDEF_UND_OCCURS_KAT FOREIGN KEY
( NR ) REFERENCES TEST.REDEF_UND_OCCURS ( NR )": "Success"}]
},
{"GroupId": "Test",
"OutputTargetName": "Postgres",
"ValidFrom": "0000-00-00-00.00.00.000000",
"SQL commands": [
{"DROP TABLE IF EXISTS \"public\".\"REDEF_UND_OCCURS_PLZ\"": "Success"
},
...
{"ALTER TABLE \"public\".\"REDEF_UND_OCCURS_KAT\" ADD CONSTRAINT \"public_REDEF_UND_OCCURS_KAT_f
k\"   FOREIGN KEY ( \"NR\" )  REFERENCES \"public\".\"REDEF_UND_OCCURS\" ( \"NR\" ) ON UPDATE CA
SCADE": "Success"}]}]}
```

The SQL commands were executed without errors.

Error messages for resource Export

If an error occurs when executing an option of resource Export, an HTTP return code not equal to 200 is returned.

In addition, there is output in the JSON data, which can contain up to three pieces of information.

“Error” always appears in the output with the problem description, e.g.

```
HTTP/1.1 400 Bad Request

{"Error": "Method 'CREAT' is not supported"}
```

In addition, the “Reason” for the problem can be displayed for some error messages, e.g.

```
HTTP/1.1 400 Bad Request


{"Error": "Export failed",
"Reason": "No parameters specified to define the export process"}
```

Finally, the actual “Exception” is displayed if such an exception was recognized in the “Reason” as the cause of a problem, e.g.

```
HTTP/1.1 500 Internal Server Error

{"Error": "Export failed",
"Reason": "Caught exception while performing BatchExport",
"Exception": "No output table found fitting the specified parameters.;"}
```

Resource Tool

This resource provides methods for maintaining the repository (initialize, backup and restore) as well as an option for starting processes.

As with import and export, this resource also requires information that makes calling the resource methods more extensive. In principle, the resource requires the same parameters that the batch utility 'BatchTool' also requires. The possible parameters and their meaning can therefore be found in the manual tcV7Batch_Components_en.pdf.

Note: However, the -connect parameter (and in the case of an SSL connection also the -keystore parameter) can be ignored. Normally the BatchTool requires these in order to establish a connection to the Agent holding the repository. Here automatically the Agent is used that is defined for starting the REST API server.

The parameters -file, -action and -overwritefile are also not required and will therefore be ignored.

INIT Tool -> Initialize the repository

The repository is initialized using the INIT option. In principle, this option does not require any additional parameters. The only exception: if the repository resides in a Db2 database, then the parameter -repoinitdb2info is required to specify the database name and/or tablespace.

The output provides information about the outcome of the initialization.

Example of initializing the repository:

The repository should be initialized. No parameters are needed for this.

```
curl -iX INIT http://127.0.0.1:8080/Tool
HTTP/1.1 200 OK

{"Success": "Initializing repository finished successfully."}
```

The repository was initialized successfully.

BACKUP Tool -> Creating a repository backup

Using the BACKUP option, a backup of the repository is created.

A complete backup of the entire repository is created without any further parameters. By specifying the -groupid parameter, the full backup can be restricted to a group.

Instead of a full backup, a partial backup can also be created. To do this, the backup is restricted at an input table level using the -tablepattern parameter. A regular expression must be specified as the parameter value. The backup then contains all input tables that match the regular expression, all output tables that are connected to these input tables, as well as all other repository objects that are somehow dependent on the tables (fields, connections, ...). Notes on regular expressions can also be found in the tcV7Batch_Components_en.pdf manual.

The output is in JSON notation. It includes all lines of the backup which are contained in a JSON array called “Statements”. These can be saved by the caller and used for later possible restores of the repository.

Example of a partial backup of the repository:

The Oracle input table “ORCL.DEMO.ARTIKEL” (and therefore also all connected output tables) of group “Test” should be saved.

These requirements result in the following backup parameters:

`tablepattern=^Test\.ORACLE\.DEMO\.ARTIKEL$`

As already mentioned in the import resource, the special characters used must be encoded accordingly, which results in the following query string:

`tablepattern=%5ETest%5C.ORACLE%5C.ORCL%5C.DEMO%5C.ARTIKEL%24`

```
curl -iX BACKUP "http://127.0.0.1:8080/Tool?tablepattern=%5ETest%5C.ORACLE%5C.ORCL%5C.DEMO%5C.AR
TIKEL%24"
HTTP/1.1 200 OK

{"Statements": [
"*tcV:RCF001",
"*TRG:V7024",
"*SRC:POSTGRESQLhost=192.168.0.22;port=5432;dbname=postgres;user=postgres;password=xxx;",
"*TYP:UPDATE",
"*REM:Partial backup date 2024-02-29-07.41.49.146000",
"*SYS:CONSWARNING         =N",
"*SYS:INITIALIZED         =2022-06-02-13.03.58.753000",
"*SYS:LASTBACKUP          =2022-05-09-16.04.25.000000",
"*SYS:LIKE_WC             =%",
"*SYS:LOCK                =READY: 2024-02-28-13.51.25.203060",
"*SYS:MODULO              =((%s % %s) = %s)",
"*SYS:VERSION             =7024",
"*STM:W*0050000173DELETE FROM <SCHEMA><CREATOR>InputTables where GroupId = <PMARKER> AND SourceT
ype = <PMARKER> AND SourceName = <PMARKER> AND ValidFrom = <PMARKER> AND ValidUntil = <PMARKER>0
000545200011452000064520001745200026452000026NTestNORACLENORCL.DEMO.ARTIKELN2020-08-12-13.07.45.0
00000N9999-99-99-99.99.99.999999",
"*STM:W*0050001145INSERT INTO <SCHEMA><CREATOR>InputTables (GroupId, SourceType, SourceName, Val
idFrom, ValidUntil, Description, AgentName, AliasSourceType, AliasSourceName, ImporterVersion, D
efaultCCSID, DblDefaultCCSID, UniqueKeyFields, DLIConcatKeyLen, DLISegmentKeyStart, DLISegmentKe
yLen, DLISegmentSeq, DLISegmentLen, DLISegmentParent, IDMSControlLen, IDMSAftGetRout, VSAMType,
VSAMRecordFormat, VSAMKeyLen, VSAMKeyPos, VSAMLogstreamId, DATACOMElements, DATACOMKeyName, DB2T
ableName, RecordExitFields, RecordExitCode, DLISegmentVariable, AdabasPassword, AdabasCiphercod
e, DB2TableVersion, DLISSA, AdabasIsnOffset, ProjectName, BulkRecordCount, BRCDate, AdabasLOBFil
eNumber, InternalUse) values (<PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>,
<PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKE
R>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMA
RKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>,
<PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKE
R>, <PMARKER>) 0004245200011452000064520001745200026452000264480000045200015452000004520000004520
0007496000114960001144800000496000114960001149600011496000114960001145200000496000114520000004520
0000452000014960001149600011452000004480000044800000452000004480000044800000452000014520000004520
000049600011448000004960001145200000496000114520002649600011452000000NTestNORACLENORCL.DEMO.ARTIK
ELN2020-08-12-13.07.45.000000N9999-99-99-99.99.99.999999YNtcAgent-WindowsYYN0101002N00000001252N
00000001200YN00000000000N00000000000N00000000000N00000000000N00000000000YN-0000000001YYNFN000000
00000N00000000000YYYYYYNNYYN00000000000YN00000000000YN00000002836N2021-02-25-08.14.45.132406N000
00000000Y",
...
"*STM:W*0050000580INSERT INTO <SCHEMA><CREATOR>ReplicationLinkage (GroupId, InputSourceType, Inp
utSourceName, OutputTargetName, OutputTableName, InputFieldName, OutputFieldName, InputValidFro
m, InputValidUntil,OutputValidFrom, OutputValidUntil, Description, ConvertCondType, ConvertCondP
arm1, ConvertCondParm2, ConvertCondParm3, ConvertCondScope, ConvertProcType, ConvertProcParms) v
alues (<PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>,
<PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>, <PMARKE
R>, <PMARKER>, <PMARKER>)00019452000114520000645200017452000084520001445200005452000054520002645
20002645200026452000264480000049600011448000004480000044800000452000014960001144800000NTestNORAC
LENORCL.DEMO.ARTIKELNPostgresNpublic.artikelNPREISNPREISN2020-08-12-13.07.45.000000N9999-99-99-9
9.99.99.999999N2021-01-07-14.24.33.158000N9999-99-99-99.99.99.999999YN0000000000YYYNRN000000000
01Y",
"*tcV:EOF"]}
```

The backup commands have been created successfully.

Alternative call:

As already mentioned in previous resources, the parameters can alternatively be sent in the request data instead of the query data. To do this, the parameters must be stored in a file in JSON format, and this file is then sent in the request data. The advantage of this method is that the call is clearer and the above-mentioned encoding of the special characters in the JSON file can be omitted.

The content of the JSON file, e.g. “PartBackupParms.json”, would be as follows in the example above:

```
{"tablepattern": "^Test\.ORACLE\.ORCL\.DEMO\.ARTIKEL$"}
```

The backup itself then reduces to the following call:

```
curl -d @PartBackupParms.json -H "@auth" -iX BACKUP http://127.0.0.1:8080/Tool
```

Example of a full repository backup:

All repository objects assigned to group “Test” should be backed up.

These requirements result in the following backup parameter:

groupid=Test

```
curl -iX BACKUP http://127.0.0.1:8080/Tool?groupid=Test
HTTP/1.1 200 OK

{"Statements": [
"*tcV:RCF001",
"*TRG:V7024",
"*SRC:POSTGRESQLhost=192.168.0.22;port=5432;dbname=postgres;user=postgres;password=xxxx;",
"*TYP:INIT",
...
"*tcV:EOF"]}
```

The backup commands have been created successfully.

Alternative call:

Again, the parameters can alternatively be sent in the request data instead of in the query data. To do this, the parameters must be stored in a file in JSON format, and this file is then sent in the request data. The advantage of this method is that the call is clearer and the above-mentioned encoding of the special characters in the JSON file can be omitted.

The content of the JSON file, e.g. “FullBackupParms.json”, would be as follows in the example above:

```
{"groupid": "Test"}
```

The backup itself then reduces to the following call:

```
curl -d @FullBackupParms.json -H "@auth" -iX BACKUP http://127.0.0.1:8080/Tool
```

RESTORE Tool -> Restoring a repository backup

Using the RESTORE option, an existing backup of the repository is restored.

Several parameters must be taken into account for a restore.

When restoring a complete backup, -groupid can be used to specify which group the contained repository objects should be restored for.

Restoring a complete backup would completely overwrite the existing repository. To prevent accidental overwriting, the -overwriterepo parameter has to be set.

Likewise, when restoring a complete backup, the components to be restored can be restricted using the -componentstorestore parameter.

These parameters and their meaning can be found in the tcV7Batch_Components_en.pdf manual.

The actual commands for the restore are transferred in the request data using “--data-binary @File_with_Data” in the call.

The output contains either a success message or, in the event of an error, the error message from the database, which made the processing stop.

Example of a repository restore:

A full backup should be restored into the repository. Only the entries from group “Test” are wanted; any entries related to system monitoring contained in the backup are not. The data for the restore is sent with the request data.

These requirements lead to the following parameters:

```
groupid=Test
componentstorestore=ADOEPJH
overwriterepo=Y
file=REQUESTDATA
```

The data file “RestoreData.txt” contains the commands generated by a backup call:

```
*tcV:RCF001
*TRG:V7024
*SRC:ORACLE    DSN=//192.168.0.1:1521/ORCL;UID=system;PWD=encxxxx;;CREATOR=TEST
*TYP:INIT
*REM:Partial backup date 2023-12-11-14.53.26.964000
*SYS:INITIALIZED         =2023-11-30-14.28.25.468000
*SYS:LIKE_WC             =%
*SYS:LOCK                =READY: 2023-12-08-14.25.32.214442
*SYS:VERSION             =7024
*STM:W*0050000133INSERT INTO <SCHEMA><CREATOR>Groups (GroupId, ValidFrom, ValidUntil, Descriptio
n) values (<PMARKER>, <PMARKER>, <PMARKER>, <PMARKER>)000044520000745200026452000264480000 0NTest
N0000-00-00-00.00.00.000000N9999-99-99-99.99.99.999999Y
...
*tcV:EOF
```

```
curl --data-binary @RestoreData.txt -iX RESTORE „http://127.0.0.1:8080/Tool?groupid=Test&compone
ntstorestore=ADOEPJH&overwriterepo=Y&file=REQUESTDATA“
HTTP/1.1 200 OK

{"Success": "Restoring repository entries finished successfully."}
```

The restore was executed successfully.

START Tool -> Starting a process

The START option can be used to start a process (defined in the repository).

The output about the outcome of the action only provides information about the start, but not about the outcome of the process.

Example of starting a process:

The process “BULK_DB2_2_PG_DEMO_ARTIKEL” defined in group “Test” is to be started. Only 50 data records should be transferred and an error trace should be written.

These requirements lead to the following parameters for the process start:

```
groupid=Test
processname= BULK_DB2_2_PG_DEMO_ARTIKEL
processparm=PM_I.MAX_INPUT_RECORDS=50 PM_I.Trace='ERRORS'
```

As already mentioned in previous resources, the special characters used must be encoded accordingly, resulting in the following query string:

```
groupid=Test&processname=BULK_DB2_2_PG_DEMO_ARTIKEL&processparm=PM_I.MAX_INPUT_RECORDS%%3D50%%20
PM_I.Trace%%3D'ERRORS'
```

```
curl -iX START "http://192.168.0.22:8080/Tool?groupid=Test&processname=BULK_DB2_2_PG_DEMO_ARTIKE
L&processparm=PM_I.MAX_INPUT_RECORDS%3D50%20PM_I.Trace%3D'ERRORS'"
HTTP/1.1 200 OK
```

```
{"Success": "Starting process finished successfully."}
```

The process was started successfully.

Alternative call:

Again, the parameters can alternatively be sent in the request data instead of in the query data. To do this, the parameters must be stored in a file in JSON format, and this file is then sent in the request data. The advantage of this method is that the call is clearer and the above-mentioned encoding of the special characters in the JSON file can be omitted.

The content of the JSON file, e.g. “StartProcParms.json”, would be as follows in the example above:

```
{"groupid": "Test",
"processname": "BULK_DB2_2_PG_DEMO_ARTIKEL",
"processparm": "PM_I.MAX_INPUT_RECORDS=50 PM_I.Trace='ERRORS'"}
```

The process start itself is then reduced to the following call:

```
curl -d @StartProcParms.json -H "@auth" -iX START http://127.0.0.1:8080/Tool
```

Error messages for resource Tool

If an error occurs when executing an option of resource Tool, an HTTP return code not equal to 200 is returned.

In addition, there is output in the JSON data, which can contain up to three pieces of information.

“Error” always appears in the output with the problem description, e.g.

```
HTTP/1.1 400 Bad Request

{"Error": "Method 'BACKUPP' is not supported"}
```

In addition, the “Reason” for the problem can be displayed for some error messages, e.g.

```
HTTP/1.1 400 Bad Request

{"Error": "Tool failed",
"Reason": "Parameter '-processname=' is missing but needed to start a process"}
```

Finally, the actual “Exception” is displayed if such an exception was recognized in the “Reason” as the cause of a problem, e.g.

```
HTTP/1.1 500 Internal Server Error

{"Error": "Starting process",
"Reason": "Caught exception while performing BatchTool",
"Exception": "The specified process 'BULK_DB2_2_PG_DEMO_ARTIKEL' in group 'Tast' does not exis
```

```
t.; "
```

Resource Migrate

This resource provides an option to migrate the RDRS meta information from one repository to another.

The migration takes place with input tables as a starting point. This means that all input tables that are to be migrated are copied from the source to the target repository. This copy process includes all fields of the table, all output tables connected to this input table, including their fields, all connections between the input and output fields of the tables, as well as all other repository objects belonging to the tables (SQL histories, additional information to IDMS, DLI, Adabas, ...).

As with the previous resources, this resource also requires information that makes calling up the resource's methods more extensive. In principle, the resource requires the same parameters that the batch utility 'BatchMigrate' also requires. The possible parameters and their meaning can therefore be found in the manual *tcV7BatchMigration_en.pdf*.

START Migrate -> Migrating the repository

Using the START option, the contents of one repository are migrated to another.

The objects to be migrated can be restricted using additional parameters and a file containing regular expressions Adjustments can be made to the data of the repository to be migrated in order to adapt the resulting new repository to the conditions of a new environment (e.g. other databases, other schema names, ...).

The output is in JSON notation. It provides information about the outcome of the migration process and contains the contents of the log files. Each of these log files is a JSON array called “Goodlog”, “Badlog” or “Trace” and contains the content of the corresponding log file.

Example of migrating a repository:

The content of a source repository is to be migrated to a target repository. Only objects from group “Test” should be taken into account and should be assigned to group “DEFAULT” in the target repository. All three log types (“GOOD”, “BAD” and “TRACE”) should be returned. All objects in the target repository should be overwritten if they already exist there; only existing Output Targets and exit codes should be excluded. Adjustments to the new environment are controlled using the regular expression file “regex.txt”, which is passed using request data (--data-binary @regex.txt).

These requirements lead to the following migration parameters:

```
reposource=user1/pwd1@192.168.0.1:5120
repotarget=user2/pwd2@192.168.0.2:5120
groupsource=Test
grouptarget=DEFAULT
goodlog=Y
badlog=Y
trace=Y
forcerenewall=Y
excludeexistingoutputtargets=Y
excludeexistingexitcodings=Y
regexfile=REQUESTDATA
```

As already mentioned in previous resources, the special characters used must be encoded accordingly, resulting in the following query string:

```
reposource=user1%2Fpwd1%40192.168.0.1:5120&repotarget=user2%2Fpwd2%40192.168.0.2:5120&groupsourc
e=Test&grouptarget=DEFAULT&goodlog=Y&badlog=Y&trace=Y&forcerenewall=Y&excludeexistingoutputtarge
ts=Y&excludeexistingexitcodings=Y&regexfile=REQUESTDATA
```

```
curl --data-binary @regex.txt -iX START "http://127.0.0.1:8080/Migrate?reposource=user1%2Fpwd1%4
0192.168.0.1:5120&repotarget=user2%2Fpwd2%40192.168.0.2:5120&groupsource=Test&grouptarget=DEFAUL
T&goodlog=Y&badlog=Y&trace=Y&
forcerenewall=Y&excludeexistingoutputtargets=Y&excludeexistingexitcodings=Y&
regexfile=REQUESTDATA"
HTTP/1.1 200 OK

{"Goodlog": [
"09:39:54:092 Starting good log",
"09:39:54:093 Checking program parameters.",
"09:39:54:100 Connection successful to Source Repository Agent 'Agent-Windows-1'",
"09:39:54:101   Host version:  7.0.0",
"09:39:54:101   Host revision: 1888",
"09:39:54:105   Repo version:  7024",
"09:39:54:117 Connection successful to Target Repository Agent 'Agent-Windows-2'",
"09:39:54:117   Host version:  7.0.0",
"09:39:54:117   Host revision: 1888",
"09:39:54:120   Repo version:  7024",
"09:39:54:124 ",
...
"09:40:58:536 Migrate finished with RC 0",
"09:40:58:536 Batch Migration finished successfully."
],
"Badlog": [
"09:39:54:093 Starting bad log",
"09:39:54:093 Checking program parameters.",
"09:39:54:100 Connection successful to Source Repository Agent 'Agent-Windows-1'",
"09:39:54:101   Host version:  7.0.0",
"09:39:54:101   Host revision: 1888",
"09:39:54:105   Repo version:  7024",
"09:39:54:117 Connection successful to Target Repository Agent 'Agent-Windows-2'",
"09:39:54:117   Host version:  7.0.0",
"09:39:54:117   Host revision: 1888",
"09:39:54:120   Repo version:  7024",
"09:39:54:124 ",
...
"09:40:58:536 Migrate finished with RC 0"
],
"Trace": [
"09:39:54:093 Starting trace",
"09:39:54:093 Checking program parameters.",
"09:39:54:093 Connecting to Agent for Source Repository.",
"09:39:54:100 Connection successful to Source Repository Agent 'Agent-Windows-1'",
"09:39:54:101   Host version:  7.0.0",
"09:39:54:101   Host revision: 1888",
"09:39:54:105   Repo version:  7024",
"09:39:54:105 Connecting to Agent for Target Repository.",
"09:39:54:117 Connection successful to Target Repository Agent 'Agent-Windows-2'",
"09:39:54:117   Host version:  7.0.0",
"09:39:54:117   Host revision: 1888",
"09:39:54:120   Repo version:  7024",
...
"09:40:58:536 Migrate finished with RC 0",
"09:40:58:536 Disconnecting from Agent for Source Repository.",
"09:40:58:537 OK."]}
```

The migration was completed successfully and the contents of the log files were returned.

Error messages for resource Migrate

If an error occurs when executing an option of resource Migrate, an HTTP return code not equal to 200 is returned.

In addition, there is output in the JSON data, which can contain up to three pieces of information.

“Error” always appears in the output with the problem description, e.g.

```
HTTP/1.1 400 Bad Request

{"Error": "Method 'STARRT' is not supported"}
```

In addition, the “Reason” for the problem can be displayed for some error messages, e.g.

```
HTTP/1.1 400 Bad Request

{"Error": "Migrate failed",
"Reason": "No parameters specified to define the migration process"}
```

Finally, the actual “Exception” is displayed if such an exception was recognized in the “Reason” as the cause of a problem, e.g.

```
HTTP/1.1 500 Internal Server Error

{"Error": "Migration failed.",
"Reason": "Error connecting to Agent. Message received:",
"Exception": "java.net.ConnectException: Connection refused: connect"}
```

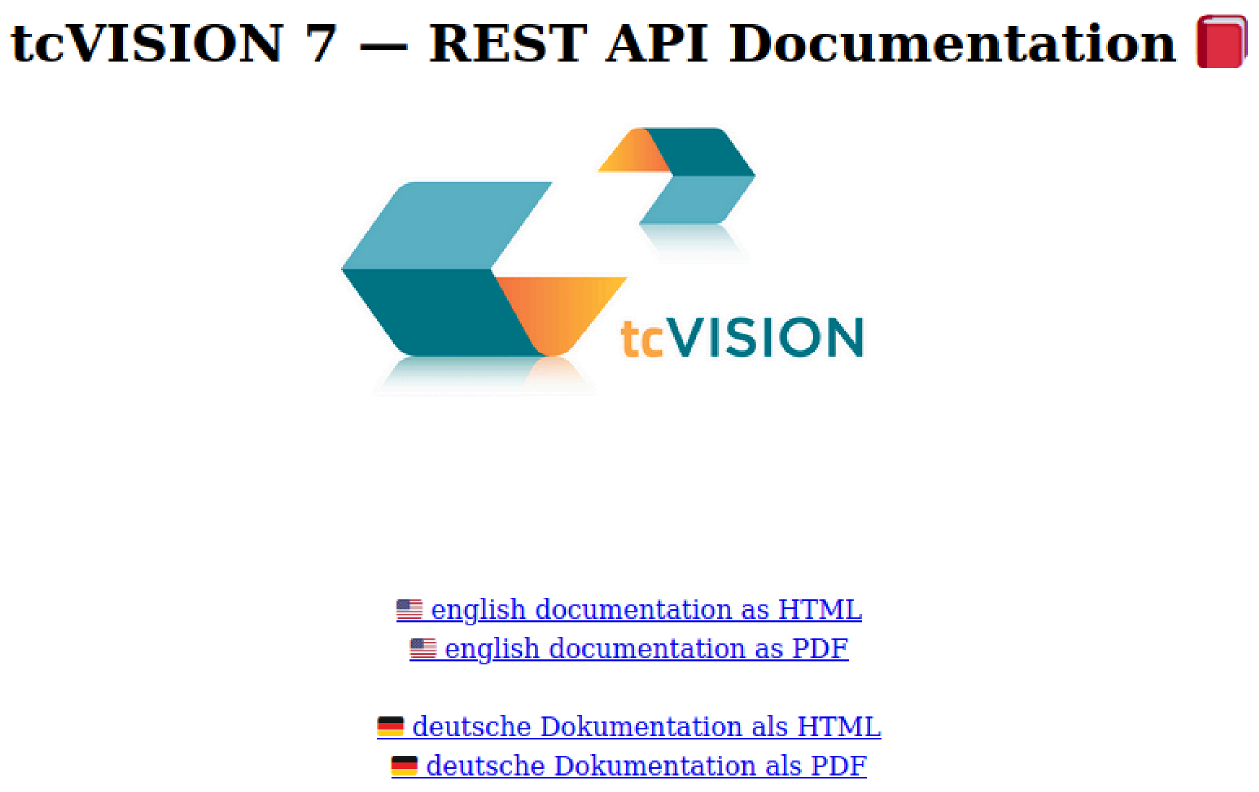
If the migration process fails (connection error, database error, ...), it is also recommended to take a look at the “TRACE” log file, as it contains a detailed error message (including possible exceptions). In the event of an error the contents of the log files are also included in the output to make it easier to investigate the cause.

Resource (online documentation)

GET / -> Forwarding to the online documentation

REST API’s online documentation will be opened in an internet browser by typing IP address and port of the REST API in the browser’s address bar. The documentation can be viewed in English or German, as PDF or in HTML format.

```
http://127.0.0.1:8080
```



tcVISION 7 — REST API Documentation 📕

tcVISION

🇺🇸 english documentation as HTML
🇺🇸 english documentation as PDF

🇩🇪 deutsche Dokumentation als HTML
🇩🇪 deutsche Dokumentation als PDF

Such a browser call uses the HTTP method `GET`. Invoked via curl, a HTML document forwarding to the URL of the online documentation is returned:

```
 curl -iX GET http://127.0.0.1:8080/
HTTP/1.1 200 OK

<!DOCTYPE html><html><head><meta http-equiv="refresh" content="0;url='https://www.bos-digitec.co
m/n_tcvision/restapihelp/index.html'" /></head><body><p>Please follow <a href="https://www.bos-d
igitec.com/n_tcvision/restapihelp/">this link</a>.</p></body></html>
```

Thus, the online documentation’s URL is:

https://www.bos-digitec.com/n_tcvision/restapihelp

Security

HTTPS

Without special settings, REST API calls occur via the HTTP protocol.

Via option `-https_keystore`, a KeyStore file needs to be set for the HTTPS protocol to be activated. For security reasons, HTTP gets deactivated, thus only HTTPS calls are possible. If it is intended that HTTP is activated nevertheless, the option `-https_only=n` must be set.

The default port for HTTPS is 8443. For another port to be chosen, it needs to be specified via `-https_port`.

An example for such a call via HTTPS:

```
curl -H "@auth" -iX GET https://127.0.0.1:8443/Agents/defined/
```

`-https_protomin=#` states the lowest level of the SSL protocol that is to be used. On successful handshake, the highest protocol both sides are offering is used. If not set, all from SSL V3 to TLS1.3 is offered.

`-https_protomax=#` states the highest level of the SSL protocol that is to be used. On successful handshake, the highest protocol both sides are offering is used. If not set, all from the lowest level to TLS1.3 is offered.

`#` denotes a number from 1 to 5, which correspond to the protocol versions SSLv3, TLSv1, TLSv1.1, TLSv1.2 and TLSv1.3.

Without specification of lowest or highest protocol level, merely the highest two are activated, TLSv1.3 und TLSv1.2.

Optionally, `-https_provider` states a provider. If no one is set, the default provider is used. For provider handling, at least OpenSSL version 3.0.0 must be used.

By each REST API call, the following information is written to the trace (if enabled):

Protocols:

If lowest or/and highest level is stated, for example `-https_protomin=3`:

```
Manually enabled SSL protocols: [TLSv1.1, TLSv1.2, TLSv1.3]
```

If not:

```
Originally enabled SSL protocols: [TLSv1.3, TLSv1.2]
```

The provider that is taken:

```
Used provider: SunJSSE
```

Authentication

In a production environment, it is highly advised to invoke the REST API solely in an authenticated manner. The authentication can be activated for Rocket® Data Replicate and Sync Agents in tcSCRIPT 7's configuration file `tcAgent.ini`. Details on this can be read in the manual `tcV7Agent_Security_en`.

The REST API gets started without authentication. For each HTTP method call, stating user name and password is essential if authentication is enabled. This is done via header. Using curl, REST API calls look like this, for example:

```
curl -iX GET -H username:<user name[@domain]> -H password:<password> http://127.0.0.1:8080/RestApi
```

With such a call, the password may and should be given in encrypted form for security reasons. It is more convenient to specify the header as a file using `@`:

```
curl -H "@auth" -iX GET http://127.0.0.1:8080/RestApi
```

where *auth* is a plain text file with the content:

```
username:<user name[@domain]>
password:<password>
```

Here, too, the password should be entered in encrypted form. Alternatively, it can also be specified unencrypted and the file made readable only for the owner of the file using the Unix/Linux command `chmod 400 <file>` .

If this header information is not provided at all, incorrectly or incompletely with activated authentication, the execution of the HTTP method is aborted, status code HTTP/1.1 401 Unauthorized and an error is output in the form of a JSON object (with details depending on the security setting):

```
[
{"ERROR": "Error connecting to agent. Received message: Error 0x08100005 occurred during communi
cation to the Agent. The process cannot be completed.",
"Last returncode": "08100005",
"Returncode explanation": "A LDAP error occurred. The username/password is incorrect."}
]
```

Access to resources

For certain method calls, access to resources authorization is required if security is enabled in the Rocket® Data Replicate and Sync Agent. For instance, exiting the REST API (`STOP /RestApi`) is only accomplishable, if the access right `control` for the profile `SYSTEM.SHUTDOWN` is permitted to the currently connected user. Details on this are described in the manual `tcV7Agent_Security_en`.

Important: It is not shown, if user permissions are not granted during a REST API call, provided that this does not result in an error. For instance, the method call `GET /Processes/active` will not list running processes (even if existent) if a user has no reading right for the profile `LIST.PROCESSES`. The HTTP status code is `200 OK` as well.

401 Unauthorized - Error due to missing access rights

If missing access rights result in an error, HTTP status code 401 Unauthorized and an error message (security settingsdependent) is issued.

Like that, the REST API can not be terminated if the access right is not granted (see above):

```
 curl -H "@auth" -iX STOP http://127.0.0.1:8080/RestApi
HTTP/1.1 401 Unauthorized
```

```
[
{"ERROR": "A LDAP error occurred. The given user could not be found or has not been permitted fo
r this action.",
"Status": "REST API still running"}
]
```

Copyright

License information on third-party components used can be found in Copyright