Octo IP
User Manual

RDS Routing Server

OCTO IP User Manual V3.1

February 2024

Your audio. Our solution.

2wcom

Manufacturer

2wcom Systems GmbH • Am Sophienhof 8 • 24941 Flensburg • Germany
Phone (+49) 461 662830-0 • Fax (+49) 461 662830-11
contact@2wcom.com • www.2wcom.com

About this Manual

Please keep this user manual for future reference.

You can download the latest version of this user manual here:

https://download.2wcom.com/products/OCTO_HZDA/

Contents

1 About the OCTO IP

The OCTO IP is a software for Windows that operates as a highly adaptable routing center that enables you to efficiently centralize inputs and outputs of RDS data streams, simplifying network management. Offering a comprehensive overview of your network's status at a glance, it is a valuable tool for broadcasters seeking reliable and failsafe operation. Whether you require centralization, redundancy, or comprehensive logging, this routing server is your trusted partner in maintaining a resilient and efficient broadcasting infrastructure.

OCTO IP operates by utilizing TCP/IP for all communication and routing. A failsafe server structure must be ensured with unique IP addressing and port addresses, for example a cluster under Windows servers. GPI/O is also managed through TCP/IP. The protocol is based on UECP, and it can encapsulate data in XML frames for specific tasks.

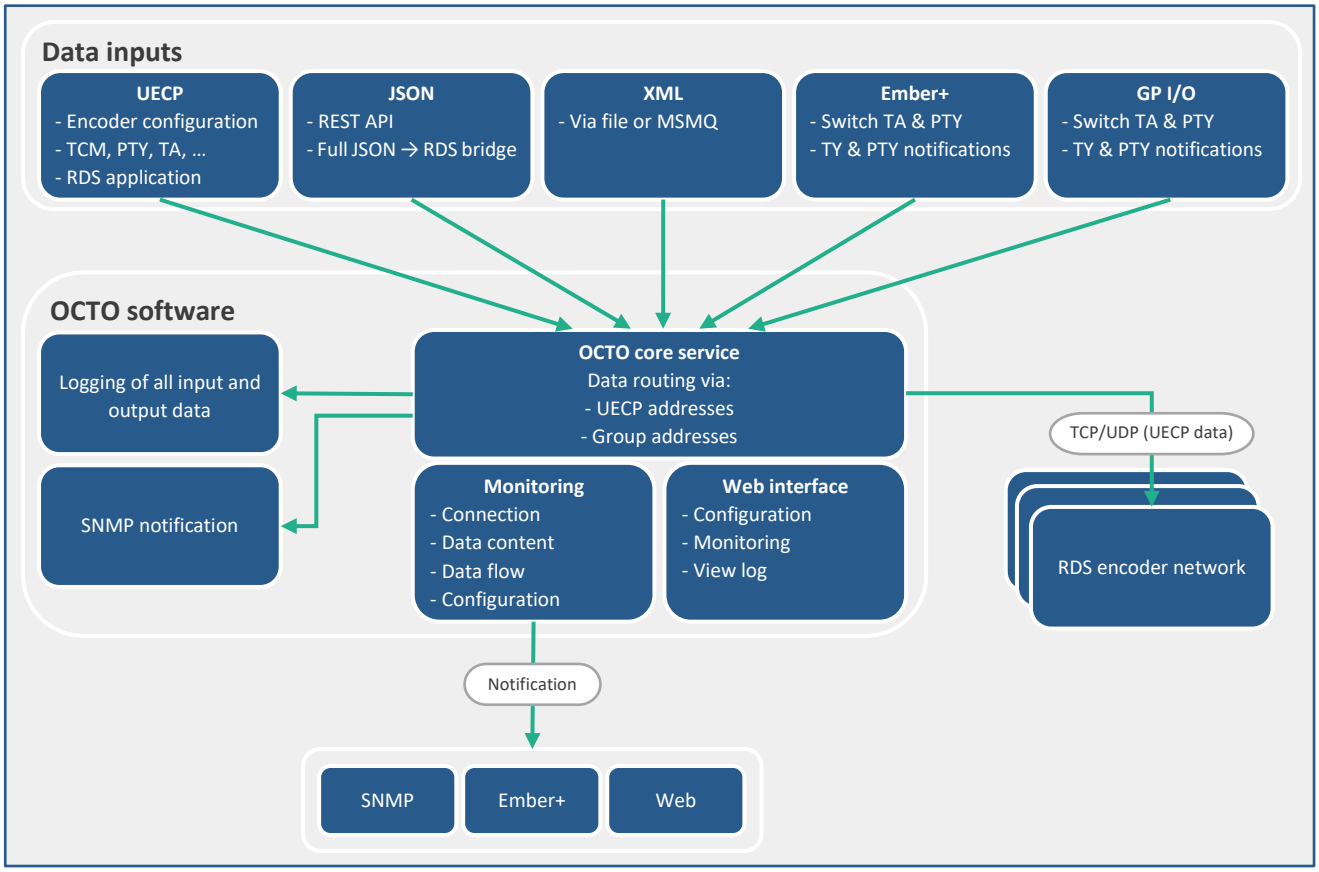
Input versatility: The OCTO IP supports the input of multiplexed data and XML data over IP and outputs it as UECP data streams to the individual RDS encoders. It is compatible with various protocols, including UECP, GP I/O, and Microsoft Message Queues.

User-friendly operation: You can make all configurations in a comprehensible web interface where you can set up easy routing from inputs to outputs by defining human-readable names for groups of encoders.

Logging and monitoring: The software provides extensive logging and monitoring capabilities to keep track of network activities and diagnostics. It supports SNMP, Ember+, and web notifications as part of the monitoring function.

Priority notifications: OCTO IP features special handling of traffic announcements (TA) and emergency warnings over PTY, ensuring that essential information is delivered effectively.

Scalability: The system accommodates a great number of connections, and you have the flexibility to purchase and use just as many as you need, given that the substantial hardware resources are available.



2 Installation

2.1 System Requirements

System Requirements	
Operating system	For running the OCTO IP software on a system, a PC with Windows server is presumed. The package runs as a service under operating system control without your intervention, i.e. it can be configured to start automatically at system startup. This way of operation is suited for a clustered server farm as well.
Disk space	The disk space needed for the OCTO IP software with the optional software Encoder-Check is less than 5 MB.
Privileges	The software package needs access to the different communication ports, i.e. TCP/IP must be ensured and serial RS-232C lines if applicable. Write access must be granted for logging.
Directory structure	For operating the OCTO IP server software, a normal directory structure is assumed, i.e. the operating system is installed on drive C: , and e.g. data and log files are stored to drive D:.

2.2 Installing the OCTO IP Software

You can install the OCTO IP software either as an application or as a service.

The OCTO application comes with an old user interface designed for 64-bit Windows. It also starts a web server that you can use to access the web interface. The advantage of using the application is that it is easier to install. Please note that the older Windows user interface may not support all OCTO functions, so it is recommended to use the web service for configuration and testing.

Installing the OCTO service requires the use of a shell (cmd) with administrator privileges. Like the OCTO application, the OCTO service does also provide a web service for further configuration.

To install the OCTO IP as an application:

1. Extract the provided folder to a location of your choice.
2. In the application folder, execute the *OCTO_IP_App.exe* file.

⇒ The app is now installed. It's recommended to use the web interface for further configuration.

To install the OCTO IP as a service:

1. Extract the provided folder to a location of your choice.
2. Open the Windows command line (cmd).
3. Find the folder where the OCTO package is located, and enter:
   `C:\OCTO_v2.00\service>` **`OCTO_IP_Service.exe -install`**
   (assuming your installation folder is *OCTO_v2.00\service*\:)
   - ➢ The Windows Service Panel now displays the registered new OCTO service:

4. Start the service using the context menu of the entry.

⇒ The service is now installed. Use the web interface for further configuration.

If the application or service fails to start or displays an error message, you may require a Microsoft VC++ Runtime Package. You can download it directly from Microsoft:
Latest supported Visual C++ Redistributable downloads | Microsoft Learn

Using either the service or the app version, the OCTO will provide a web service for further configuration. The default port of the web service is:

- 8000 for http
- 8001 for https
- 9000 for Ember+

You can edit the value of the ports in the *OCTO_IP.ini* file inside the *config* folder.

For the https service, we use a default, self-signed certificate that you can replace if necessary. The certificate is located in the root of the installation folder: *octo.pem.* In case you change the values, the service must be restarted.

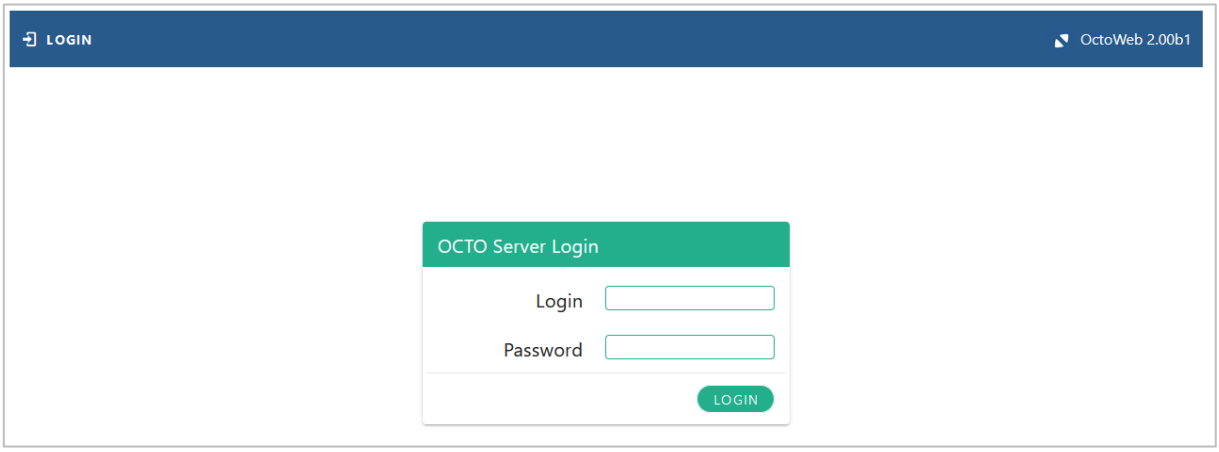
3 General Operation

3.1 Logging in to the Web Service

To open the OCTO web service, use the following URL:

- http://your-pcs-ip-address:8000/
- https://your-pcs-ip-address:8001/

The login page appears:



Enter your login credentials. The default login credentials are:

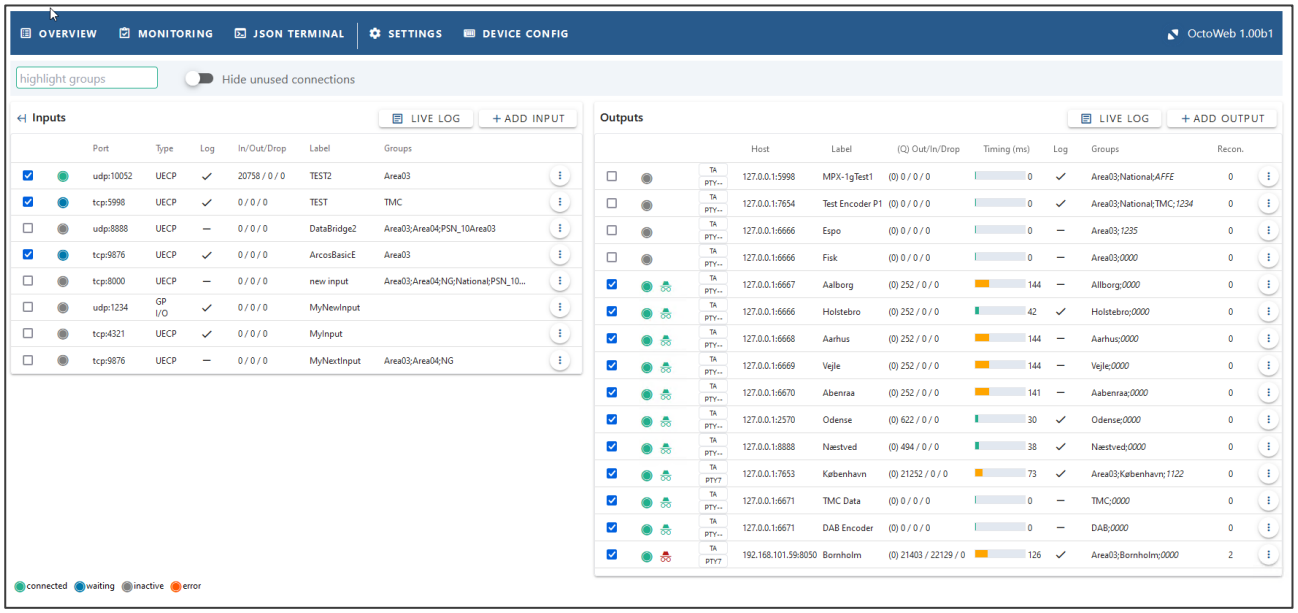
- admin / admin
- operator / operator
- monitor / monitor

The login credentials can be edited in the *OCTO_IP.ini* file.

After the first login, the **Monitoring** page is empty because the monitoring parameters have not been set up, yet. On the **Overview** page, displays default inputs and outputs to get started with.

3.2 Viewing the Overview Page

The **Overview** page shows the inputs and outputs.



Parameter	Description
Port	shows the selected input connection.
Type	shows the selected input type (UECP, MMQ, GP I/O).
Log	shows whether logging is enabled.
In/Out/Drop	shows a frame count of (frames in wait loop), received frames, sent frames, and dropped frames.
Host	shows the selected connection for output (IP address + IP port or COM port)
Label	shows the defined input label.
Groups	shows the selected group(s).
Timing	shows the turnaround time of a data packet. If this value is consistently high, it is advisable to review and assess the system configuration.
Recon.	shows the reconnection count.

The color code gives information on the state of the connections:

connected waiting inactive error

- Unselect the checkbox of an input or output that you do not want to use at the moment.
- To only show selected inputs and outputs, toggle the **Hide unused connections** switch.
- Click on the button to hide the inputs and only show the outputs.
- Use the search bar to highlight inputs and outputs with specific groups.

You can do the following tasks on the **Overview** page:

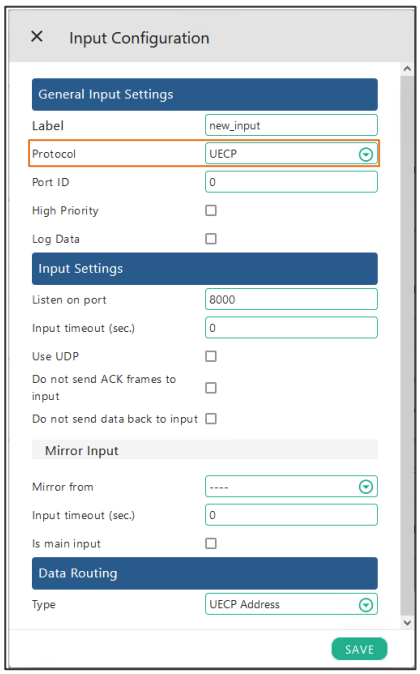
- Adding and Editing an Input
- Adding and Editing an Output
- Activating or Deactivating an Input or Output
- Viewing the Live Log of an Input or Output
- Setting up Monitoring of an Output

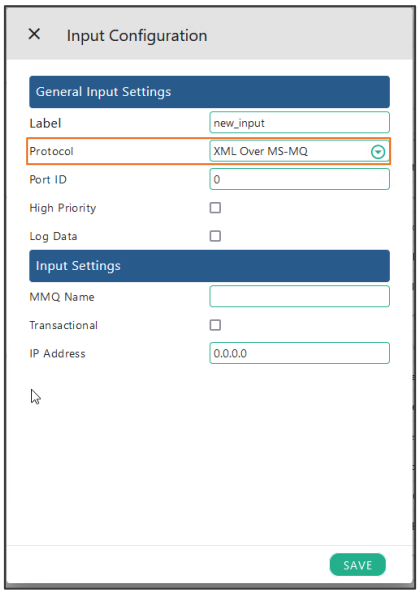
For more detailed information on the status of the inputs and outputs, see 4.2 “Viewing the Monitoring Page”.

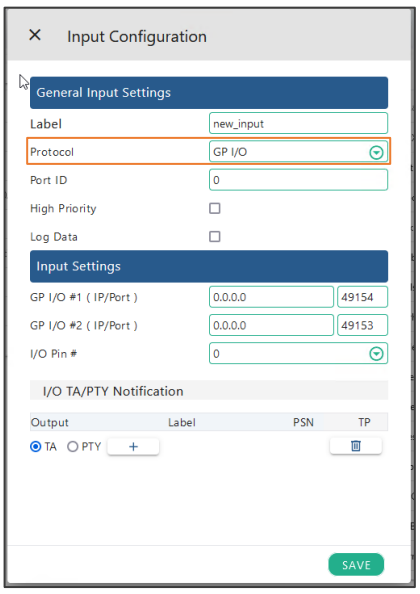
3.3 Adding and Editing an Input

The OCTO IP server allows multiple simultaneous connections (queues) to your system management server. These connections are differentiated based on whether they use the IP network, with different port numbers, or GP I/Os, with distinctions based on pin numbers. For each port, you can have a separate queue. The queues from multiple ports can be combined and directed to a single output. This approach offers the advantage of faster processing because the input queues run in parallel, while the commands and actions within a single queue are processed sequentially. Furthermore, XML input is automatically converted into UECP format before processing. The input for XML is based on Microsoft Message Queues (MMQ).

1. Go to the **Overview** page.
2. To create a new input entry, select **Add Input**.
3. To edit an existing entry, select **Edit** in the entry’s menu. Alternatively, double-click on the entry.
4. In the **Input Configuration** window, configure the settings. The configurable settings depend on the selected protocol type and are explained below.
5. Select **Save**.







Settings for UECP

Settings	Description
Label	Specify the name of this input for better reference.
Protocol	Select UECP.
Port ID	Enter the Port ID for this input.
High Priority	Activate the checkbox to ensure that the data of this input makes it to the output, even in case of data output overload.
Log Data	Log the data received on this input. Hourly, the log is locally stored as a non-size-limited hex-dump file.
Listen on Port	Enter the IP port to be used.
Input Timeout	Enter the IP port timeout.
Use UDP	Select this checkbox to use UDP instead of UECP.

Do not send ACK frames to input	Select this checkbox to not send acknowledgment frames (ACKs) back to the input source. ACK frames are used to confirm the receipt of data. Disabling this feature means that the system will not acknowledge the reception of data from the input.
Do not send data back to input	Select this checkbox to not send data back to the input source.
Mirror input	When enabled, this feature allows you to mirror or duplicate the input data for specific purposes. Specify the source input from which you want to mirror the data. Define a timeout period for this mirrored input. This is the duration for which the mirrored data will be active. Select whether this input is the main source of data.
Type	Select the data routing type.

Settings for XML over MS-MQ

Settings	Description
Label	Specify the name of this input for better reference.
Protocol	Select XML over MS-MQ.
Port ID	Enter the Port ID for this input.
High Priority	Activate the checkbox to ensure that the data of this input makes it to the output, even in case of data output overload.
Log Data	Log the data received on this input. Hourly, the log is locally stored as a non-size-limited hex-dump file.
MMQ Name	Enter a short name for the XML over MMQ input.
Transactional	Activate the checkbox if the MMQ service requires transactional data.
IP Address	Enter the IP address of the server with the MMQ service.

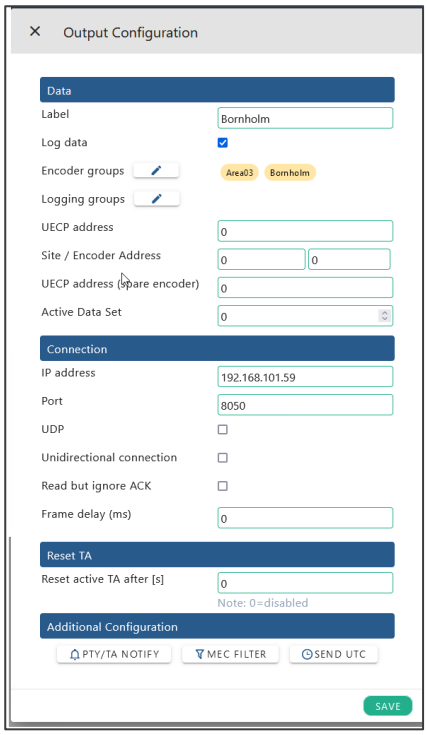
Settings for GP I/O, NC06, and Ember+

Settings	Description
Label	Specify the name of this input for better reference.
Protocol	Select GPI/O.
Port ID	Enter the Port ID for this input.
High Priority Input	Activate the checkbox to ensure that the data of this input makes it to the output, even in case of data output overload.
Log Data	Log the data received on this input. Hourly, the log is locally stored as a non-size-limited hex-dump file.
GP I/O #1	Enter the IP address of the first GPIO and the corresponding IP port.
GP I/O #2	Enter the IP address of the backup GPIO (uses the same I/O pin #).
I/O Pin #	Select the GP I/O pin to be used.
TA/PTY	Activate **TA** or **PTY** if you want the GP I/O pin to trigger a TA/PTY notification. Use the **Add** button (+) to configure the destination of the notification. Use the **Delete** button (🗑) to delete a TA/PTY notification entry.

3.4 Adding and Editing an Output

Up to 128 output / communication ports can be configured to operate over TCP/IP. Each IP output has a different IP address with an assigned port number. For every type of operation, ensure that the software has privileges and rights to use the communication interfaces.

1. To create a new output entry, right-click on the right list. Select **Add Output**.
2. To edit an existing entry, double-click on the entry.
3. In the window **Output Configuration**, configure the settings. They are explained below.
4. Select **Save**.



Output Configuration Settings

Settings	Description
Label	Specify the name of this output for better reference.
Log data	Log the data received on this output. The log is locally stored hourly as a non-size-limited hex-dump file.
Encoder groups	Select the necessary routing groups. For PSN routing, select the PSN groups that correspond to the PSN of your encoders. OCTO_IP then scans the incoming UECP data for a PSN that matches a configured PSN group name and routes it via the configured IP address or COM port to the corresponding encoder. These group names need to be configured in the *OCTO_IP.ini* file (see section 6.5 OCTO_IP.ini).
Logging groups	Select the necessary logging groups. The logging groups (for example *LogData1*) can be used to forward the UECP data to separately configured outputs.
UECP address	Enter the UECP address of the encoder in hex code.
Site / encoder address	Enter the UECP address, split up in site and encoder number (in hex). This is an alternative input method to **UECP address**.
UECP address (spare encoder)	Enter the UECP address of the spare encoder.
Active data set	Enter the active data set for proper rerouting of dynamic data to the spare encoder.
IP address	Enter the IP address of the output destination.
Port	Enter the IP port number of the output destination.
UDP	Select this checkbox to use UDP for the IP connection instead of TCP (this automatically selects the **Unidirectional connection** checkbox is activated automatically).
Unidirectional connection	If this checkbox is selected, there is only data flow from an input to the output (no responding traffic). The only exceptions are acknowledge frames that are sent by OCTO IP to the sending application on the input, if PSN/Group routing is used while one input is routed to more than one output.
Read but ignore ACK	Select this checkbox to have the OCTO IP read acknowledge frames as described above.
Frame delay	Optionally induce a frame delay to prevent buffer overflow.
Reset active TA after [s]	Enter a delay time to activate the TA reset.

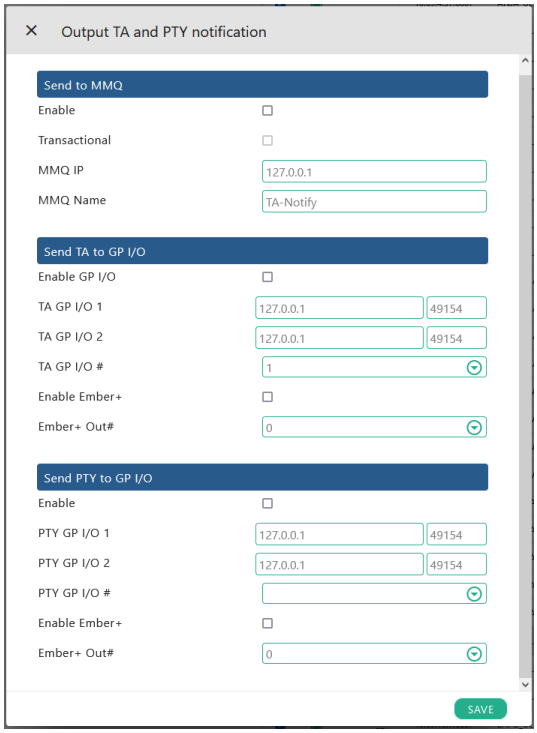
Send UTC	Select the **Send UTC** button and configure the settings for transmitting the current local time to the encoder at specified intervals.

PTY/TA Notify

The PTY/TA Notify function enables the signaling of Traffic Announcements (TA) and Program Type (PTY31) into your system management server. This information is defined in XML format, as specified in the provided appendix. The XML data can also work seamlessly with Microsoft Message Queues when operating as an output.

For GPO, you can specify two GP I/O modules, or a specific Ember+ output (see also 5.2.5 Ember+ Interface) for setting the defined output pin signaling TA or PTY31. The GPOs are handled over TCP/IP with optional hardware from 2wcom.

NOTE: When the OCTO IP sends TAs to the encoders, they will instantly put the TA on air (they have a socket connection, i.e. TCP). The XML packages sent to your system management server may be processed asynchronously, and they are queued in case the connection is temporarily disrupted.



Settings	Description
Send to MMQ / Transactional / MMQ-IP / MMQ-Name	For Microsoft Message Queues with outgoing TAs and PTY31, an IP address and a queue name are required to send the appropriate XML message, transactional or non-transactional.
Send TA to GP I/O **TA GP I/O 1:** **TA GP I/O 2:** **TA GP I/O #:**	For GPO, two GPI/O modules can be used and specified (enter the IP address in “TA GP I/O 1:” and/or “TA GP I/O2:”) for setting the defined output pin (TA GP I/O #) signaling TA. In case TA is treated, the appropriate encoder for signaling is the one holding the Traffic Program (TP), which is always exactly one per TX station (it is not possible to define more than one encoder/output-queue switching the same output-pin).
Send PTY to GP I/O **PTY GP I/O 1** **PTY GP I/O 2** **PTY GP I/O #**	For GPO two GPI/O modules can be used and specified (enter the IP address in the “PTY GP I/O 1:” and/or “PTY GP I/O2:” field) for setting the defined output pin (PTY GP I/O #) signaling PTY31.
Ember+	Select the checkbox to send the notification with Ember+. Select the number of the Ember+ output from the dropdown menu.

MEC Filter

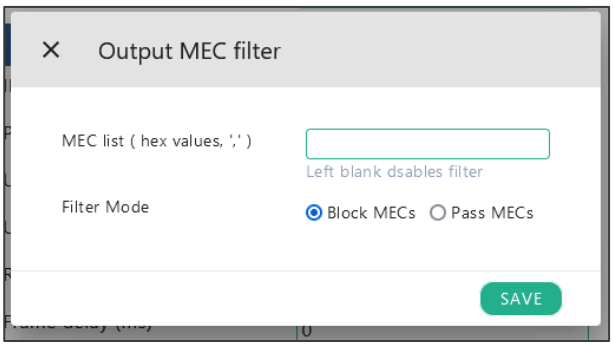
The MEC filter allows you to control which specific MEC commands are permitted to pass through or are blocked from passing through the OCTO.

To set up the MAC filter:

1. Open the output settings.
2. Select **MEC Filter**.
3. Select the filter mode to either block or only allow the specific MECs.
4. In the field **MEC list**, enter the hex values of the MEC commands to filter.
   Example: 01,02,0A (MEC for PI,PS,RT)
   Refer to the UECP specification for information on all possible MEC commands.
5. Select **Save**.

⇒ If the MEC filter has blocked a MEC command, it is logged in the corresponding error log.

NOTE: Use the filter function only if required for your application and make sure to configure the MEC filter carefully. To avoid operating errors, note that the MEC filter for each individual output is enabled until you completely blank the MEC list input field and click on **Save**.



3.5 Activating or Deactivating an Input or Output

To activate an input or output:

▶ Select the checkbox of the specific input or output entry.

⇒ The color of the corresponding virtual LED shows the state of the connection:

- **blue:** Tool tries to connect.
- **green:** Tool established connection successfully.
- **red:** Tool cannot establish connection.
- **TA tag on output symbol:** TA flag of the corresponding RDS encoder is activated.

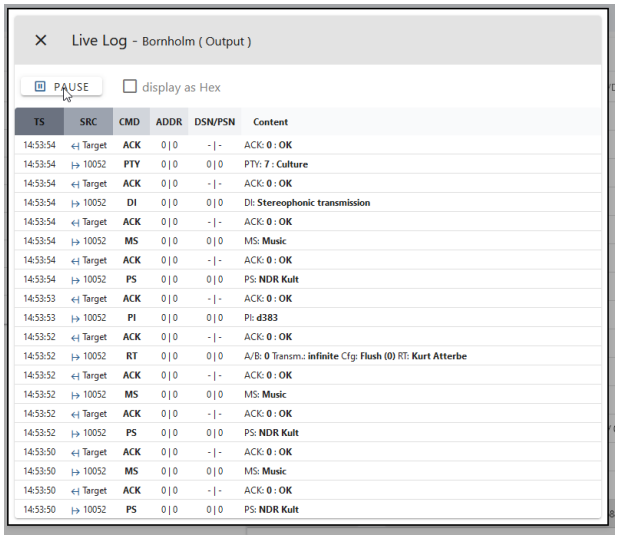
3.6 Viewing the Live Log of an Input or Output

To view the live log of an output:

1. Select the menu ⋮ of the specific input or output entry of which you want to see the port log.
2. Select **Live Log**.

⇒ The **Live Log** window shows the UECP data of this input or output.

Select **display as Hex** to view the commands as hex values.
Select **Pause** to keep new entries from appearing in the live log.

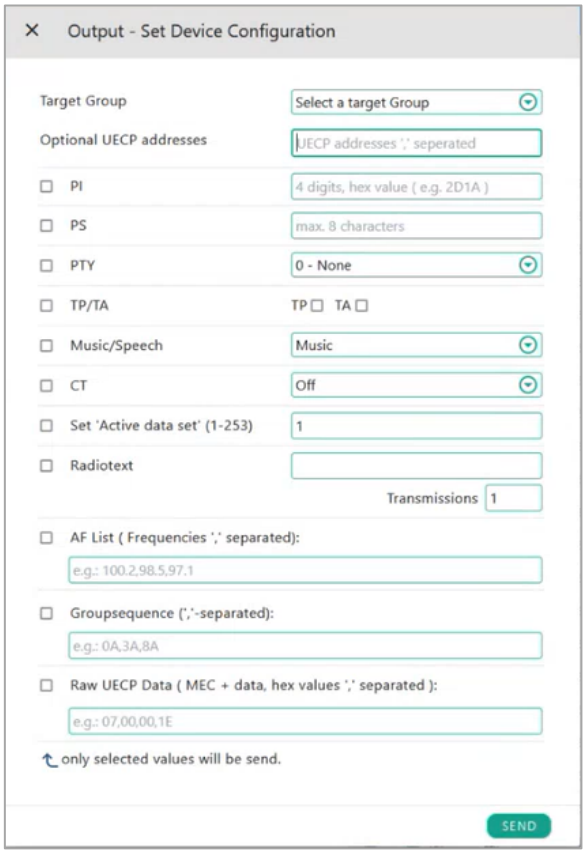


3.7 Sending RDS Data

With OCTO IP, you can send the following RDS information directly from the web interface:

- PI, PS, PTY, TP/TA, Music/Speech, CT, RT, AF

Additionally, it is possible to configure the encoders groups sequence or send any UECP command to one encoder or to a group of encoders.



To send RDS data:

1. Select **Device Config** from the navigation bar.
   - ➢ The **Output – Set Device Configuration window** opens.

2. Select the target group of encoders.
3. Configure the settings of the RDS data you want to send. Only selected values will be sent.
4. Select **Send**.

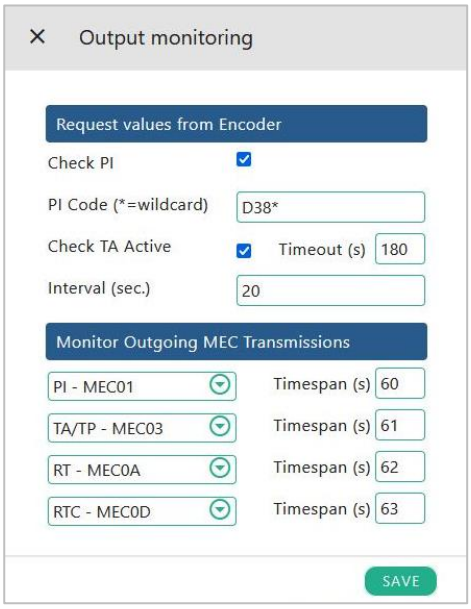
⇒ The OCTO IP transmits every single selected item in one message into the specified queue to your system management server.

4 Monitoring the System

4.1 Setting up Monitoring of an Output

It is possible to setup various monitoring parameters for each output. There are 2 different monitoring methods:

- Monitor RDS Encoder parameters (TA and PTY) read out directly from the encoder.
  This method monitors the TA and PTY that is currently on air.
- Monitor and inspect the outgoing data stream for the occurrence of specific MECs.
  This method monitors the data that leaves the OCTO. Please note that it is not guaranteed that this data is received by the encoder.



To set up monitoring of an output:

1. Select the **menu** button ⋮ of the output entry for which you want to set up monitoring.
2. Select **Monitoring**.
   ➢ The **Output monitoring** window opens.
3. Configure the settings for monitoring this output:

Check PI	Select this checkbox to request the on-air PI value from the RDS.
PI Code	Define the valid PI code. You can use an asterisk (*) as a wildcard character. This can be useful in case you use regionalization. If the requested value does not match the valid PI code, an alarm is raised.
Check TA active	Select this checkbox to monitor the timespan in which the TA is active on an output.
Timeout	Define the timeout. If the active timespan of the TA exceeds the given timeout, an alarm is raised.
Interval	Define the interval in which the parameters are read from the RDS encoder. The information read from the RDS encoder may be displayed delayed by this interval.
Monitoring specific MECs	Define up to 4 MECs. The outgoing data steam is scanned for the occurrence of the MEC. In case this MEC is not seen within the configured timespan, an alarm is raised.

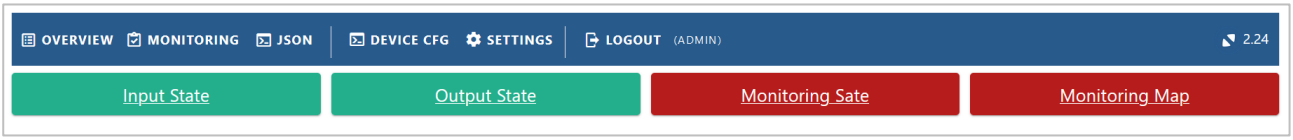
4. Select **Save**.

⇒ The monitoring function for this output is now active.

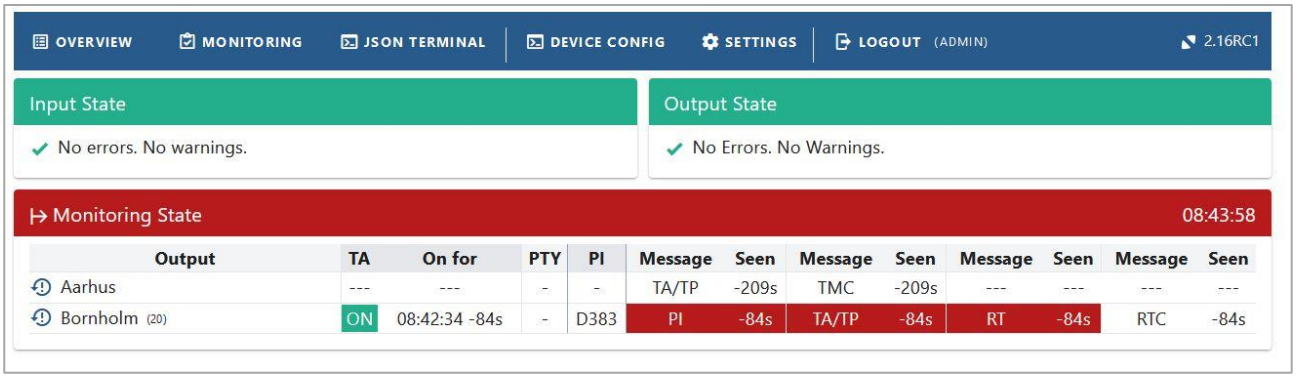
If monitoring is active for an output, it is indicated by the **spy** icon. A green **spy** icon signals normal operation. A red icon signals that at least one monitoring value is in alarm state.

4.2 Viewing the Monitoring Page

The **Monitoring** page displays a compact summary of all configured monitoring options.



The four items **Input State**, **Output State**, **Monitoring State** and **Monitoring Map** are displayed with a green background when the state is OK, with a red background when there is a problem. Click on the item of interest to see more details.



The input and output states are displayed in the **Input State** and **Output State** blocks. In case of a connection failure or a timeout the specific input or output is listed in red to indicate the alarm. Notifications such as frame drops are displayed as a note. As soon as the input or output falls back in normal operation mode the alarm condition is cleared.

The **Monitoring State** block displays the state of the configured output monitoring. Alarms are indicated by a red background. If the alarm is cleared, the background color changes back to white or green.

The following information is displayed in this block:

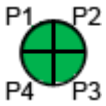
Column	Description
Output	Displays the **Label** field of the output for which the monitoring is enabled. If TA/PTY monitoring is set up, the number next to the output label indicates the next 'read parameter cycle'. If an alarm is raised, you can reset it by clicking on the R**evert** icon . This resets the monitoring timing and starts over the monitoring cycle.
TA / On for	Indicates the current state of the TA flag. If the TA state is ON, the time at which the TA state was read and the duration of the condition is displayed (for example, "-84s" indicates that the state was reached 84 seconds ago).
PTY	Displays the last seen PTY value of the output data stream. PTY31 is displayed in red to indicate an alarm state.
Message / Seen	Displays the configured MEC monitoring and the timestamp of the last occurrence of the specific MEC. If the occurrence of the MEC exceed the configured timespan, an alarm is raised. If the MEC is seen again, the alarm state is cleared.

You can use the **slide in /slide out** icon to toggle between a compact and an extended view of the **Monitoring State** block. The compact view only displays the TA, PTY, and PI values.

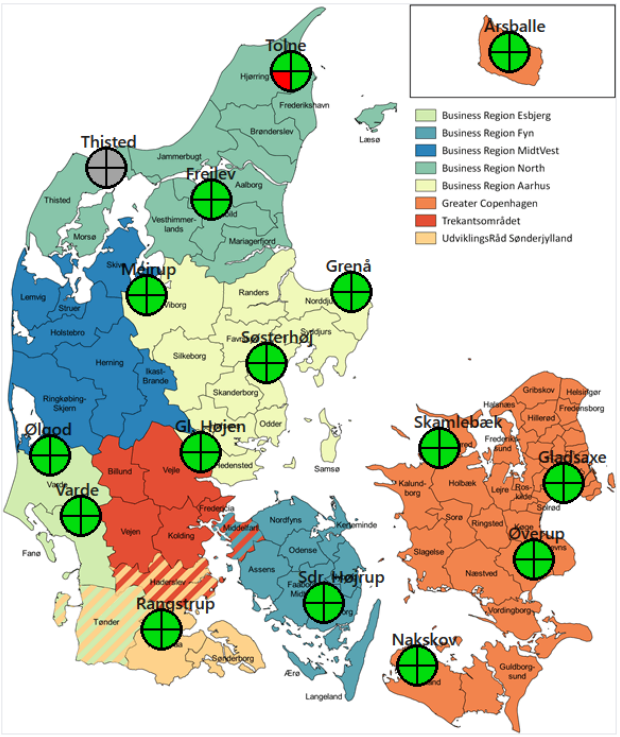
All alarm conditions (raised / clear) are also sent as an SNMP trap if it is set up.

4.3 Monitoring Overview on a Geographical Map

You can set up a map overview that displays the monitoring states. This allows you to get a quick overview of where the monitoring function detected a failure. The map can combine up to 4 encoders of one location. Each encoder is represented by a segment in the location icon:



You can place these location icons on a map image to generate a compact monitoring overview:



To set up the map view:

- **Prerequisite:** You have already set up the monitoring parameters and made sure that your monitoring works as expected.

1. Find the “html/maps/” path in the OCTO installation directory.

   It contains the icons for the monitoring states **OK**, **FAIL**, and **NONE** as well as the map image and the monitoring configuration file. The icon size is currently 48x48 pixels. You can replace the images with customized images.

   Icons for the OK state: *ok1.png, ok2.png, ok3.png* and *ok4.png*
   Icons for the FAIL state: *fail1.png, fail2.png, fail.png* and *fail4.png*
   Icons for the NONE state: *none1.png, none2.png, none3.png* and *none4.png*
   Name of the configuration file: *map_cfg.json*

2. Configure the configuration file according to your needs. Please find information on how to configure the file below.

3. Save the changes.

⇒ You have set up the map view.

Configuration File

The configuration file is stored in JSON format. It contains properties for the background map image and the display size of the map on the web page. To avoid any scaling effects, set this site to the actual size of the image.

```
"map": "map1a_dk.png",
"map_width": 578,
"map_height": 687,
```

The location information is defined in the following “location” array. One location entry defines the (display) name of the entry and a position in pixel (xpos and ypos) on the background image. The status icon will be displayed at that position. The bottom-left coordinate of the background image is (0,0).

In the example below, the location ‘Tolne’ will be displayed at position (248, 600) on the map. The map size is 578x687. Hence, the location will be displayed near the middle-top of the map.

```
    "name" : "Tolne",
    "xpos" : 248,
    "ypos" : 600,
```

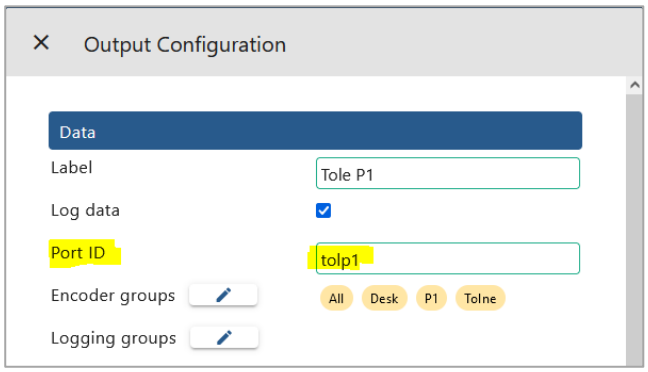
The following “targets” information, again an array, defines the relation of an output (encoder) to the state icon. The id field refers to the ‘Port ID’ field in the *Output Configuration property* dialog. The “label” is the display information of the program. The “alarms” field defines which alarm types should be considered for the status-icon. An empty “alarms” field will disable the state display for that specific item.

```
        "id": "tolp1",
        "label" : "P1",
        "alarms" : "connection,ta,pty,mec"
```

The “alarms” field can contain multiple values listed with a comma (,) as separator. Valid values are:

- connection – display the IP connection state
- ta – display ta timeout alarms
- pty – display PTY31 alarms
- pi – display PI mismatches
- mec – display ‘mec occurrence’ errors.

Reference between the ‘id’ field in the config file and the ‘Port ID’ field in the output configuration dialog:



Complete example JSON with two locations, 4 encoders each:

```
{
  "map": "map1a_dk.png",
  "map_width": 578,
  "map_height": 687,
  "location" :
  [
      {
      "name" : "Tolne",
      "xpos" : 248,
      "ypos" : 600,
      "targets" :
      [
        {
          "id": "tolp1",
          "label" : "P1",
          "alarms" : "connection"
        },
        {
          "id": "tolp2",
          "label" : "P2",
          "alarms" : "connection"
        },
        {
          "id": "tolp3",
          "label" : "P3",
          "alarms" : "connection"
        },
        {
          "id": "tolp4",
          "label" : "P4",
          "alarms" : "connection,ta,pty,pi,mec"
        }
      ]
    },
    {
      "name" : "Frejlev",
      "xpos" : 173,
      "ypos" : 479,
      "targets" :
      [
        {
          "id": "holp1",
          "alarms" : "connection"
        },
        {
          "id": "holp2",
          "alarms" : "connection"
        },
        {
          "id": "holp3",
          "alarms" : "connection,ta,pty"
        },
        {
          "id": "holp4",
          "alarms" : "connection"
        }
      ]
    }
  ]
}
```

5 Using the Rest API

5.1 HTTP server configuration

The OCTO server exposes a REST API. By default, a HTTP service is started along with the OCTO application. The default port is 5432 and can be modified in the OCTO.ini file, located in the config folder. Additionally, the IP address that the webserver should listen to can be specified.

The section looks like this:

```
[HTTP Server]
port=5432
log-data=1
ip=127.0.0.1
```

In case data logging is enabled, each request to the 'rds' route is logged in the *'log/json'* directory. The complete request and the complete response are each logged to a file.

The file format is:

dd.mm.yyyy-hh.mm.ss-**req**-*unique_id*.json	This file contains the JSON request.
dd.mm.yyyy-hh.mm.ss-**res**-*unique_id*.json	This file contains the JSON response.

The *unique ID* is identical for a pair of req/res files. It is used to identify multiple requests in the same second.

The format of the logs may look odd in a normal text editor. A JSON-friendly viewer is recommended (easiest way to inspect the logs is to open the file in Firefox, Safari or Edge)

Note: The size and amount of all files in the 'log/json' directory might grow quite fast, so it should be cleaned up by the system regularly.

The OCTO server must be restarted in case the values are changed.

5.2 API Routes

5.2.1 Liveness Probe

GET /**state/ping**

This route can be used for a liveness probe. It returns a JOSN file which contains the current timestamp of the server.

Example:

http://ip-of-octo-server:5432/api/**state/ping**

returns:

```
{
  "alive-at": "Mon Jan 16 10:01:55 2023"
}
```

5.2.2 Input State

GET /**state/input**

Returns the state of the OCTO inputs. This JSON includes the information which are listed in the OCTO-IP application on the input side (left window side). The JSON data contains a root element “inputs” which contains an array of input objects.

Each input object contains an “id” element which can be used as a reference for other commands (e.g. port configuration, logfiles, start/stop etc…)

The “enabled” element indicates that the input is ‘checked’ for action.

The “active” element indicates that the input is ready to receive data.

Example:

http://ip-of-octo-server:5432/api/**state/input**

returns:


```
{
  "inputs": [
    {
      "id": 0,
      "label": "TEST2",
      "protocol": "UECP",
      "port": 10502,
      "type": "tcp",
      "enabled": 1,
      "active": 1,
      "in-count": 0,
      "out-count": 0,
      "drop-count": 0
    },
    {
      "id": 1,
      "label": "TEST",
      "protocol": "UECP",
      "port": 5999,
      "type": "tcp",
      "enabled": 1,
      "active": 1,
      "in-count": 0,
      "out-count": 0,
      "drop-count": 0
    }
  ]
}
```


5.2.3 Output State

GET /**state/output**

This route works like the state/input route but returns information about the outputs.

Example:

http://ip-of-octo-server:5432/api/**state/output**

returns:


```
{
  "outputs": [
```

```
    {
      "id": 0,
      "label": "MPX-1g",
      "target-ip": "192.168.102.79",
      "target-port": 6666,
      "type": "tcp",
      "connected": 0,
      "active": 1,
      "in-count": 0,
      "out-count": 0,
      "drop-count": 0,
      "connection-lost-count": 0,
      "queue-count": 0
    },
    {
      "id": 1,
      "label": "Minnen",
      "target-ip": "192.168.102.78",
      "target-port": 6700,
      "type": "tcp",
      "connected": 1,
      "active": 1,
      "in-count": 7,
      "out-count": 7,
      "drop-count": 0,
      "connection-lost-count": 0,
      "queue-count": 0
    }
  ]
}
```


5.2.4 RDS Data

GET **/rds**

The /rds route can be used to modify the RDS data in the system. The body of the document can contain several JSON formatted directives to alter the RDS data. The OCTO will parse the JSON data, convert it to UECP and send the data to the addressed encoder(s). A root element “commands” can contain an array of objects where each object is an RDS command.

The following RDS commands are supported:

- Program Service Name **PS**
- Program Identifyer **PI**
- Program Type **PTY**
- Traffic Announcement **TA/TP**
- Radio Text **RT**
- Alternative Frequencies **AF**
- Group Sequence **GS**
- Raw UECP data

Format:

```
{
  "commands" :
  [
    {
      "id" : unique id for this command
      "command" : "rds-command",
      "value" : "value for rds-command",
      "psn" : 0,
      "dsn" : 0,
      "areas" : "addressed-area"
    }, …
  ]
}
```

Parameters	Description
id	Contains a unique id for each command. This id is used in the reply to identify error messages for a specific command. Type: Integer Note: This parameter is optional. If it is not present the OCTO generates an id for each command.
command	Contains the RDS command (pi, ps, ta, pty, rt, ms, ct, af, gs, ds, raw) Type: String
value	Contains an additional parameter for the command. Type: String
dsn	Contains the dsn parameter that should be addressed in the RDS encoder. Type: Integer Note: this parameter is optional. In case the parameters are not present, a default value of 0 is used (RDS Encoders current program).
psn	Contains the psn parameter that should be addressed in the RDS encoder. Type: Integer or String This parameter can be either a single integer number or a string containing a comma-separated list of psn numbers (for example “3,13”). In case it is a string, a UECP frame is generated for each psn number in the list.

areas	Contains the target areas where the generated rds message should be sent to. This value corresponds to the ‘output group setting’ in the OCTO server. Several values can be used delimited by ‘;’ (semicolon). This element can also contain an array of strings object.
	Additionally, each area name can append a specific UECP address which should be used. The UECP address is appended using a ‘:’ (colon) as a delimiter.
	Type: String or Array of Strings
	
	Example #1 for valid areas:
	`"areas" : "group1;group2"`
	This will send the data to the OCTO outputs which have the group ‘group1’ or ‘group2’ assigned.
	
	Example #2 for valid areas:
	`"areas" : "group1:2344;group2"`
	This will send the data to the OCTO outputs which have the group ‘group1’ or ‘group2’ assigned. In the ‘group1’ case the data will be send using the UECP address ‘2344’.
	
	Example #3 for valid areas:
	`"areas" :`
	`[`

```
  "group1:2344",
  "group2"
]
```

Same as ‘Expamle #2’ but use an alternative array format.

- The “ta” command may include the additional value “tp”.
  Type: Integer
  Note: this value is optional. In case this parameter is missing, a default value of 1 is used.
- The “rt” command must include an “rt-data” element. It contains an array of objects which contains the radio text data which should be used.

Format:

```
"rt-data":
  [
    {
      "text" : "the radio text to set",
      "transmissions" : number-of-transmissions,
      "ab-flag" : ab-flag
    },…
  ]
```

Parameters	Description
text	Contains the radiotext that should be sent. It can be up to 64 characters. Type: String
transmissions	Contains the number of transmissions. This value affects the duration the radiotext is visible. Note: The duration of the visibility is also affected by the RDS group sequence. Type: Integer Note: This value is optional. In case this parameter is missing a default value of 2 is used.
ab-flag	Contains the value of the ab-flag of the RT command. Type: Integer Note: This value is optional. In case this parameter is missing a default value of 1 is used.

- The “af” command accepts a list of frequencies in the format:
  “100.1,98.1,101.4”
- The “gs” command accepts a list of RDS groups like:
  “0A,2A,8A”
- The “raw” command accepts a UECP command without the UECP framing:
  “07,00,00,1e” (set PTY to 300)

The reply of the “raw” command has the following format:

```
{
  "state": "sum-state-of-commands",
  "commands": [
    {
      "id": unique-id-of-command,
      "state": "state-of-command",
      "command": "conversion-command-summary"
    },…
  ]
}
```

Parameters	Description
state	“state” is a ‘sum-sate’ of all commands. In case all commands could be converted and sent to the desired target this field contains “ok”. In case at least one command had an error it contains “error”.
commands	“commands” contains an array of objects. Each object refers to one “command” from the previously sent JSON file.
ip	“id” is the unique identifier from the input JSON file or an OCTO generated unique id. Type: Integer
state	“state” is set to “ok” in case the “command” could be converted to UECP and is scheduled for transmission. Contains “error” in case the “areas” parameter contains non-existing or currently non-reachable targets. It can also contain command-specific or general error messages such as "value element is missing data". Type: String
command	“command” contains the converted UECP command with the parsed parameters, and the target areas. Type: String

Example 1:

http://ip-of-octo-server:5432/api/**rds**
with body:

```
{
  "commands" :
  [
    {
      "command" : "ps",
      "value" : "YLE One",
      "psn" : 0,
      "dsn" : 0,
      "areas" : "Test"
    },
    {
      "command" : "ta",
      "tp" : 1,
      "value" : "0",
      "areas" : "Test"
    },
    {
      "command" : "pi",
      "value" : "D328",
      "areas" : "Test"
    },
    {
      "command" : "pty",
      "value" : "7",
      "areas" :
      [
        "Test",
        "Test"
      ]
    },
    {
      "command" : "rt",
      "psn" : 0,
      "dsn" : 0,
      "areas" : "Test",
      "rt-data" :
      [
        {
          "text" : "my 1st radio text",
```

```
          "transmissions" : 4,
          "ab-flag" : 1
        },
        {
          "text" : "my 2nd radio text",
          "transmissions" : 2,
          "ab-flag" : 0
        }
      ]
    }
  ]
}
```

returns:

```
{
  "state": "ok",
  "commands": [
    {
      "id": 1,
      "state": "ok",
      "command": "ps;0;0;YLE One;Test"
    },
    {
      "id": 2,
      "state": "ok",
      "command": "ta;0;0;0;Test"
    },
    {
      "id": 3,
      "state": "ok",
      "command": "pi;0;0;D328;Test"
    },
    {
      "id": 4,
      "state": "ok",
      "command": "pty;0;0;7;Test;Test"
    },
    {
      "id": 5,
      "state": "ok",
      "command": "rt;0;0;4;1;my 1st radio text;Test"
    },
    {
      "id": 6,
      "state": "ok",
      "command": "rt;0;0;2;0;my 2nd radio text;Test"
    }
  ]
}
```

Example 2 (with errors):

http://ip-of-octo-server:5432/%20api/**rds**
with body:

```
{
  "commands" :
  [
    {
      "id" : 100,
      "command" : "ps",
      "value" : "YLE One",
      "psn" : 0,
      "dsn" : 0
    },
    {
```

```
      "id" : "110",
      "tp" : 1,
      "value" : "0",
      "areas" : "Test"
    },
    {
      "id" : 120,
      "command" : "pi",
      "value" : "D328"
    },
    {
      "id" : "130",
      "command" : "pty",
      "value” : "7"
      "areas" :
        [
          "Test",
          "Test-does-not-exist"
        ]
    },
    {
      "id" : "140"
      "command" : "rt",
      "psn" : 0,
      "dsn" : 0,
      "areas" : "Test",
      "rt-data" :
      [
        {
          "transmissions" : 4,
          "ab-flag" : 1
        },
        {
          "text" : "my 2nd radio text",
          "transmissions" : 2,
          "ab-flag" : 0
        }
      ]
    }
  ]
}
```

returns:

```
{
  "state": "error",
  "commands": [
    {
      "id": 100,
      "state": " areas element is missing data",
    },
    {
      "id": 110,
      "state": " command element is missing data",
    },
    {
      "id": 120,
      "state": " areas element is missing data",
    },
    {
      "id": 130,
      "state": "error",
      "command": " pty;0;0;7;Test;Test-does-not-exist"
    },
    {
      "id": 140,
```

```
      "state": " rt text element is missing",
    },
    {
      "id": 6,
      "state": "ok",
      "command": "rt;0;0;2;0;my 2nd radio text;Test"
    }
  ]
}
```

5.2.5 Ember+ Interface

Ember+ tree structure:

```
root/rds/data
        /state
        /ts
root/GPIOs/GP-I/GPI-00
               /GPI-01
               /GPI-02
               / ...
root/GPIOs/GP-O/GPO-00
               /GPO-01
               /GPO-02
               / ...
```

To initiate a command to the OCTO the '/root/rds' path and its components can be used:

The data element is writeable. The content uses the following syntax:

```
cmd:the_cmd;value:the_value;target:the_target[;psn:the_psn;dns:the_dsn]
```

cmd: pi,ps,ta,pty
value for pi: 4 digits pi (hex)
value for ps: up to 8 chars ps
value for ta: tp (0 or 1) / ta (0 or 1)
value for pty: 0-31 (int)

target: OCTO output group
psn: 0-253 (optional parameter; default is 0)
dsn: 0-253 (optional parameter; default is 0)

example 1: `cmd:pty;value:30;target:program1;psn:3;dsn:0`
example 2: `cmd:ta;value:1/0;target:program-ta`

After writing the `data` element, the OCTO converts the string to UECP commands and sends them to the given outputs. After the conversion, the `state` field is updated ('`ok`' or '`failed`') and the `ts` field receives the current timestamp.

To initiate a TA or PTY command or requsest the TA or PTY state, the `/root/GPIOs` path and its components can be used. The `GPI-nn` items are writeable and the state is picked up and processed by an 'Ember+' input (see also 3.3).

The `GPO-nn` elements can reflect the TA and PTY state of an Encoder (see PTY/TA Notify in section 3.4).

6 Appendix

6.1 Information on RDS Data Routing

The OCTO IP handles routing of RDS information based on UECP addresses or by group names. Additionally, it is possible to use Ember+ or GPI inputs to change TA and PTY settings.

Supported input formats are UECP, JSON and XML. For each feed, a queue is implemented that works as a buffer to handle all requests from various applications. Depending on the content and routing information given in the routing setup, the data packets are routed to the appropriate outgoing feed. This setup ensures the ability to access every encoder from every input source. The optional Ember+, GPIs and GPOs are handled with additional interpreter logic.

The provided services for routing are the following:

- Automatic insertion of the UECP address, according to the used output
- Recalculation of the sequence counter and CRC code
- TA switching, single GPI TA inputs can be used for national or regional TAs. TA will be answered by the encoder and the OCTO IP server will check the answer and send a signal back to the source of the TA
- PTY switching will be handled the same way as TA
- Priority handling of PTY31
- Diagnostic of the traffic data
- Status monitoring of all server traffic

6.1.1 Notification Acknowledgements

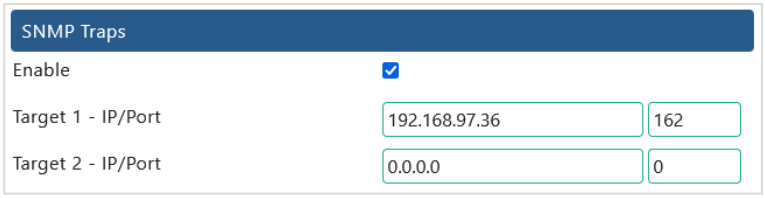
The OCTO IP server can send acknowledgments to your system management server through a Message Queue, utilizing the XML frames provided in the appendix. These acknowledgments correspond to the following events:

- TA on
- TA off
- PTY31 on
- PTY31 off

You can enable this functionality in the output settings (see *PTY/TA Notify*).

6.1.2 SNMP Traps

The OCTO IP Server can send several SNMP traps. Enable SNMP traps in the settings dialog:



The following OIDs are supported:

```
#define OCTO_INFO_EVENT_OID              "1.3.6.1.4.1.21529.10.6.0.2"
#define OCTO_TA_INFO_EVENT_OID           "1.3.6.1.4.1.21529.10.6.0.3"
#define OCTO_PTY_INFO_EVENT_OID          "1.3.6.1.4.1.21529.10.6.0.4"
```

```
#define OCTO_LI_INFO_EVENT_OID          "1.3.6.1.4.1.21529.10.6.0.5"
#define OCTO_ALERT_EVENT_OID            "1.3.6.1.4.1.21529.10.6.0.1"
#define OCTO_EVENT_OID                  "1.3.6.1.4.1.21529.10.6.1.1"
```

The following message content is delivered:

Note: Values that are enclosed in square brackets (**[** …**]**) represent a list of possible options where only one of the values is delivered. Values that are enclosed in single quotes (**'**…**'**) indicate a placeholder for variable content.

OCTO Start / Stop (OCTO_ALERT_EVENT_OID)

Is sent with each OCTO-Server start and stop.

Message content:

```
"OCTO [START|STOP]"
```

Queue Overflow (OCTO_ALERT_EVENT_OID)

Is sent when an output queue reached the configured limit.

Message content:

```
"QUEUE_OVERFLOW: Output 'port' ('label') reached queue limit ('queue_limit')"
"QUEUE_OVERFLOW_CLEAR: Output 'port' ('label') queue limit OK ('queue_limit')"
```

Input Timeout (OCTO_ALERT_EVENT_OID)

Is sent when an input gets no data for the configured timespan.

Message content:

```
"INPUT_TIMEOUT: Input 'port' ('label') reached timeout ('timeout's)."
"INPUT_TIMEOUT_CLEAR: Input 'port' ('label') timeout clear."
```

LI Change (OCTO_LI_INFO_EVENT_OID)

Is sent when the linkage introduces a regional / national switch

Message content:

```
"State Changed to [*Regional*|*National*] for - PSN:'psn_num',LI:'li_value'"
"Switched to [*Regional Mode*|*National Mode*]"
```

PTY Change - output notification (OCTO_PTY_INFO_EVENT_OID)

Is sent when the PTY on an output changes from ‘n’ to PTY31 or PTY31 to ‘n’.

Message content:

```
"PTY_CHANGE: Input:'port' ('label') PTY:'pty',DSN:'dsn',PSN:'psn'"
```

Input start/Stop (OCTO_INFO_EVENT_OID)

Output start/stop (OCTO_INFO_EVENT_OID)

Is sent everytime an input or an output is started or stopped. This trap can be disabled in the OCTO.ini file.

Message content:

```
"[Input|Output] [start|stop] 'ip' / 'label'"
```

Output connection error (OCTO_ALERT_EVENT_OID)

Is sent when an IP connection to an Encoder is lost/recovered.

Message content:

```
"Connection [lost:|recovered:] 'ip' / 'label'"
```

Monitoring state change (OCTO_ALERT_EVENT_OID)

Is sent when a monitoring item changes its state.

Message content:

```
"'ip':'port'-'label': Monitoring Alarm [ON|OFF]\t[PI|TA|PTY|MEC1|MEC2|MEC3|MEC4]
('cause')"
```

TA Change - UECP Input (OCTO_TA_INFO_EVENT_OID)

Is sent everytime an input UECP receieves a TA change.

Message content:

```
"TA_CHANGE: Input:'port' ('label') TP:'tp',TA:'ta',DSN:'dsn',PSN:'psn'"
```

PTY change - UECP input (OCTO_INFO_EVENT_OID)

Is sent every time an input UECP receives a PTY change.

Message content:

```
"'port' ('label') PTY:'pty',DSN:'dsn',PSN:'psn'"
```

6.1.3 GP I/O

The GP I/Os are handled directly over the OCTO IP server and are based on a TCP/IP socket connection. The supported hardware are the GPI/O modules and are optionally provided by 2wcom. These modules are to be wired to a panel of the broadcast service.

NOTE: It is necessary to enter the IP addresses of all 2wcom GPI/O boxes in the OCTO_IP.ini file. See section 6.5 “OCTO_IP.ini” for details.

The GP I/O function can be configured in the Input Properties and Output Properties window respectively.

6.2 XML Interchange Format

The XML interchange format between the OCTO IP server and your system management server has the following specification automatically generated by XMLSpy.

NOTE: You can enter the URLs of your own XML definitions in the OCTO_IP.ini file. See section 6.5 “OCTO_IP.ini”.

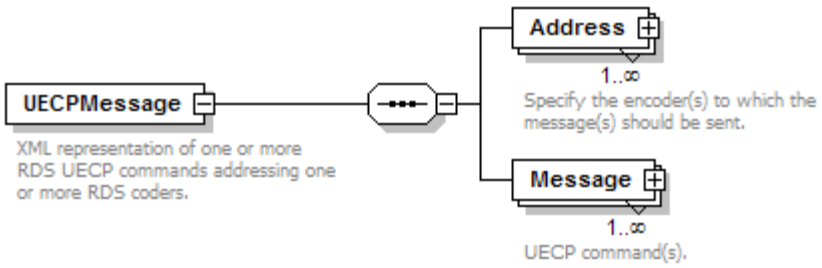
Scheme **UECPMessage.xsd**

scheme location:	**D:\Temp\UECPMessage.xsd**
attribute form default:	**unqualified**
element form default:	**unqualified**

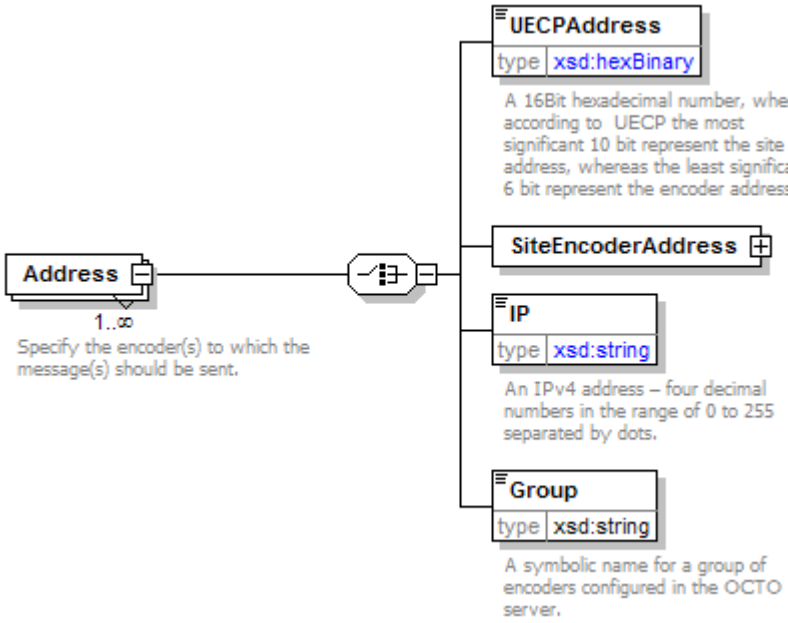
Elements

UECPMessage

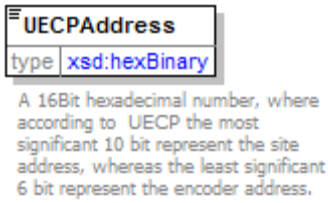
element UECPMessage

diagram	
properties	content complex
children	**Address Message**
annotation	documentation XML representation of one or more RDS UECP commands addressing one or more RDS coders.

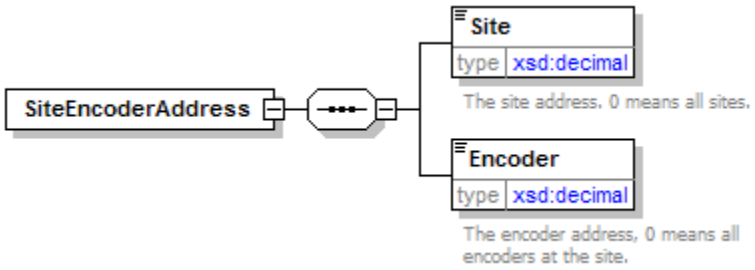
element UECPMessage/Address

diagram	
properties	isRef 0 minOcc 1 maxOcc unbounded content complex
children	**UECPAddress SiteEncoderAddress IP Group**
annotation	documentation Specify the encoder(s) to which the message(s) should be sent.

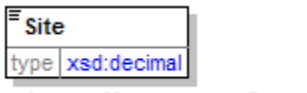
element UECPMessage/Address/UECPAddress

diagram	
type	restriction of **xsd:hexBinary**
properties	isRef 0 content simple
facets	length 2
annotation	documentation A 16Bit hexadecimal number, where according to UECP the most significant 10 bit represent the site address, whereas the least significant 6 bit represent the encoder address.

element UECPMessage/Address/SiteEncoderAddress

diagram	
properties	isRef 0 content complex
children	**Site Encoder**

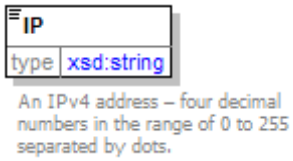
element UECPMessage/Address/SiteEncoderAddress/Site

diagram	
type	restriction of **xsd:decimal**
properties	isRef 0 content simple
facets	minInclusive 0 maxInclusive 1023
annotation	documentation The site address. 0 means all sites.

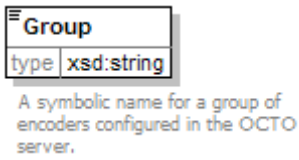
element UECPMessage/Address/SiteEncoderAddress/Encoder

diagram	
type	restriction of **xsd:decimal**
properties	isRef 0 content simple
facets	minInclusive 1 maxInclusive 63
annotation	documentation The encoder address, 0 means all encoders at the site.

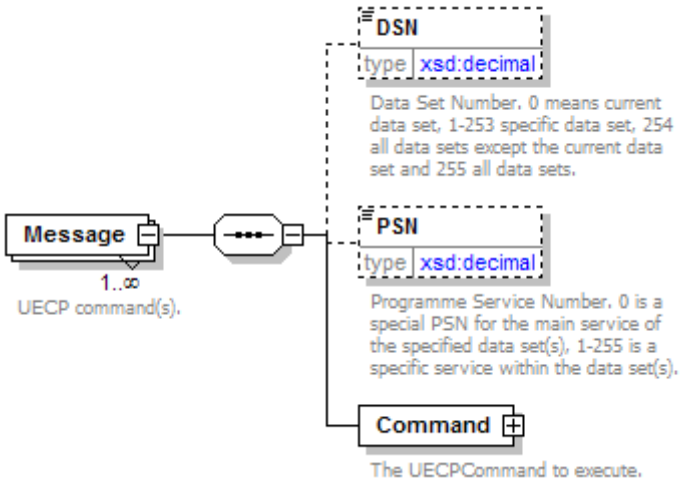
element UECPMessage/Address/IP

diagram	
type	restriction of **xsd:string**
properties	isRef 0 content simple
facets	pattern ((25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9]?[0-9])\.){3}(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9]?[0-9])
annotation	documentation An IPv4 address – four decimal numbers in the range of 0 to 255 separated by dots.

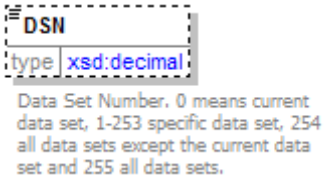
element UECPMessage/Address/Group

diagram	
type	**xsd:string**
properties	isRef 0 content simple
annotation	documentation A symbolic name for a group of encoders configured in the OCTO IP server.

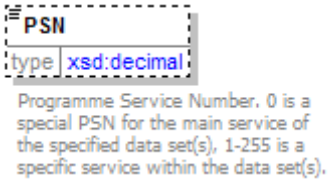
element UECPMessage/Message

diagram	
properties	isRef 0 minOcc 1 maxOcc unbounded content complex
children	**DSN PSN Command**
annotation	documentation UECP command(s).

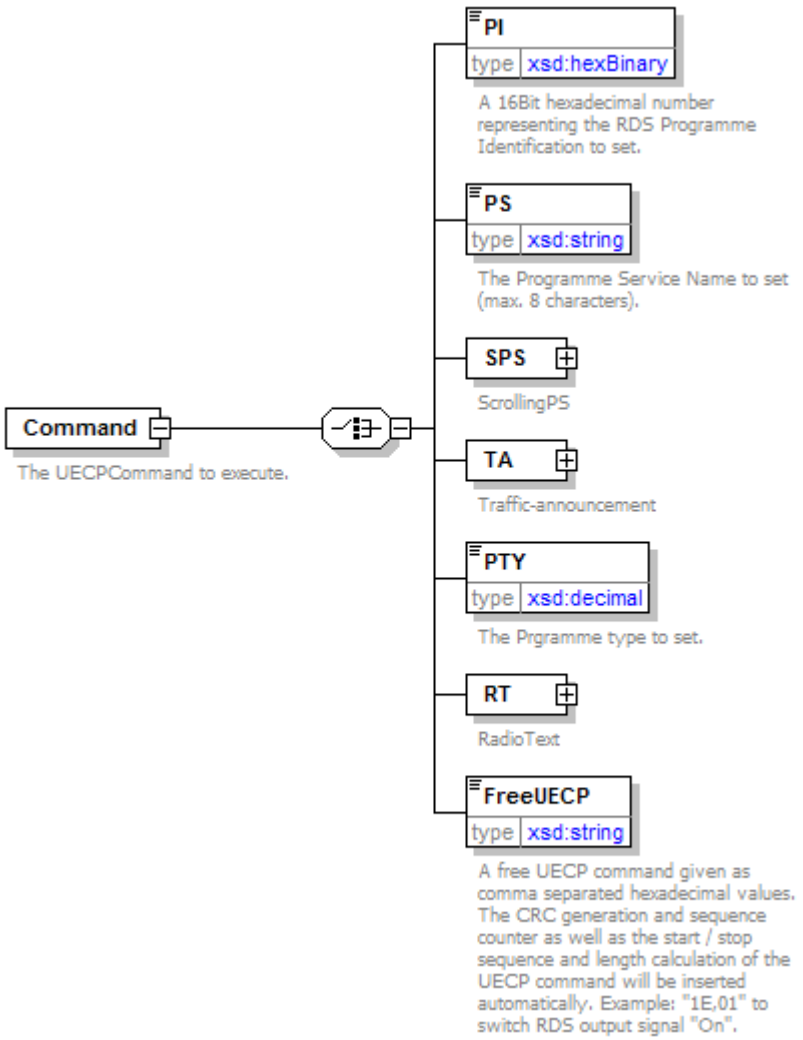
element UECPMessage/Message/DSN

diagram	
type	restriction of **xsd:decimal**
properties	isRef 0 minOcc 0 maxOcc 1 content simple default 0
facets	minInclusive 0 maxInclusive 255
annotation	documentation Data Set Number. 0 means current data set, 1-253 specific data set, 254 all data sets except the current data set and 255 all data sets.

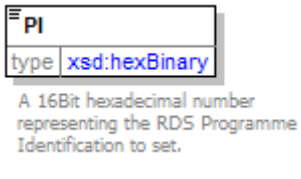
element UECPMessage/Message/PSN

diagram	
type	restriction of **xsd:decimal**
properties	isRef 0 minOcc 0 maxOcc 1 content simple default 0
facets	minInclusive 0 maxInclusive 255
annotation	documentation Programme Service Number. 0 is a special PSN for the main service of the specified data set(s), 1-255 is a specific service within the data set(s).

element UECPMessage/Message/Command

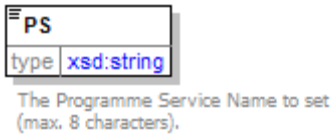
diagram	
properties	isRef 0 content complex
children	**PI, PS, SPS, TA, PTY, RT, FreeUECP**
annotation	documentation The UECPCommand to execute.

element UECPMessage/Message/Command/PI

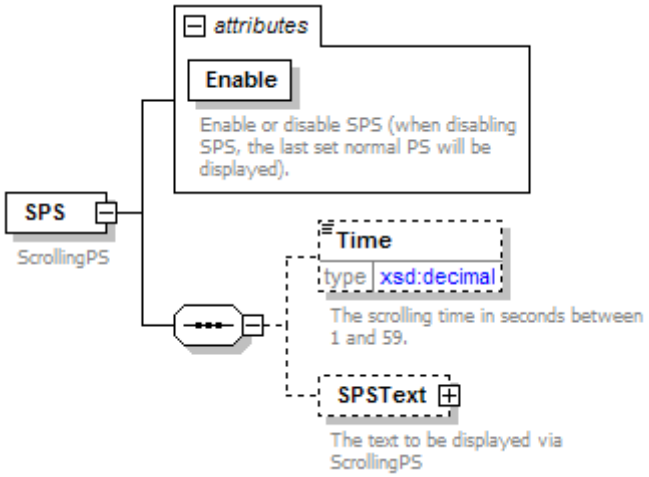
diagram	
type	restriction of **xsd:hexBinary**
properties	isRef 0 content simple
facets	length 2
annotation	documentation

	A 16Bit hexadecimal number representing the RDS Programme Identification to set.

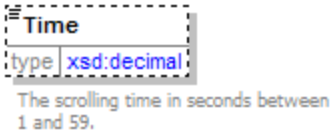
element UECPMessage/Message/Command/PS

diagram	
type	restriction of **xsd:string**
properties	isRef 0 content simple
facets	maxLength 8
annotation	documentation The Programme Service Name to set (max. 8 characters).

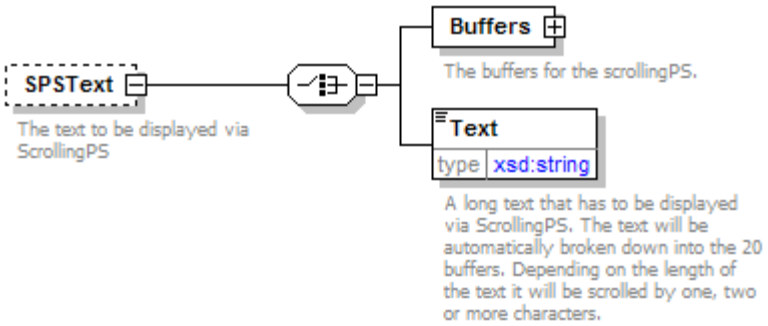
element UECPMessage/Message/Command/SPS

diagram	
properties	isRef 0 content complex
children	**Time, SPSText**
attributes	Name: Enable Type: **xsd:boolean** Use: required Default: Fixed: Annotation: documentation Enable or disable SPS (when disabling SPS, the last set normal PS will be displayed).
annotation	documentation ScrollingPS

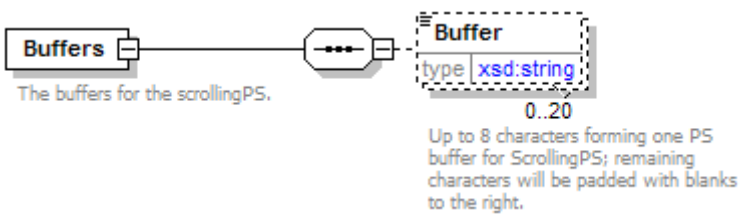
element UECPMessage/Message/Command/SPS/Time

diagram	
type	restriction of **xsd:decimal**
properties	isRef 0 minOcc 0 maxOcc 1 content simple
facets	minInclusive 1 maxInclusive 59
annotation	documentation The scrolling time in seconds between 1 and 59.

element UECPMessage/Message/Command/SPS/SPSText

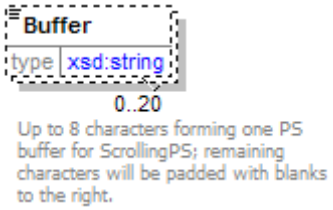
diagram	
properties	isRef 0 minOcc 0 maxOcc 1 content complex
children	**Buffers, Text**
annotation	documentation The text to be displayed via ScrollingPS

element UECPMessage/Message/Command/SPS/SPSText/Buffers

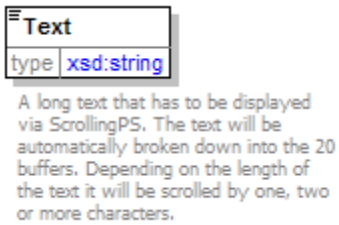
diagram	
properties	isRef 0

	content complex
children	**Buffer**
annotation	documentation The buffers for the scrollingPS.

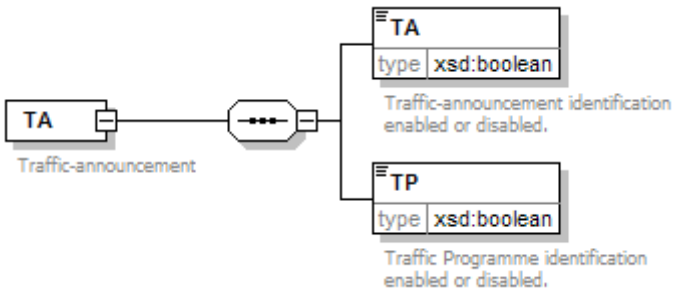
element UECPMessage/Message/Command/SPS/SPSText/Buffers/Buffer

diagram	
type	restriction of **xsd:string**
properties	isRef 0 minOcc 0 maxOcc 20 content simple
facets	maxLength 8
annotation	documentation Up to 8 characters forming one PS buffer for ScrollingPS; remaining characters will be padded with blanks to the right.

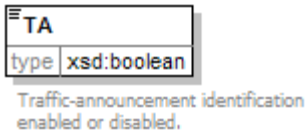
element UECPMessage/Message/Command/SPS/SPSText/Text

diagram	
type	restriction of **xsd:string**
properties	isRef 0 content simple
facets	maxLength 160
annotation	documentation A long text that has to be displayed via ScrollingPS. The text will be automatically broken down into the 20 buffers. Depending on the length of the text it will be scrolled by one, two or more characters.

element UECPMessage/Message/Command/TA

diagram	
properties	isRef 0 content complex
children	**TA, TP**
annotation	documentation Traffic-announcement

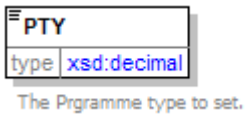
element UECPMessage/Message/Command/TA/TA

diagram	
type	**xsd:boolean**
properties	isRef 0 content simple
annotation	documentation Traffic-announcement identification enabled or disabled.

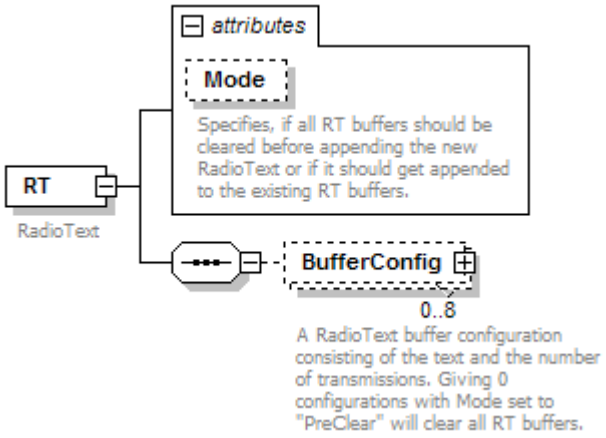
element UECPMessage/Message/Command/TA/TP

diagram	
type	**xsd:boolean**
properties	isRef 0 content simple
annotation	documentation Traffic Programme identification enabled or disabled.

element UECPMessage/Message/Command/PTY

diagram	
type	restriction of **xsd:decimal**
properties	isRef 0 content simple
facets	minInclusive 0 maxInclusive 31
annotation	documentation The Programme type to set.

element UECPMessage/Message/Command/RT

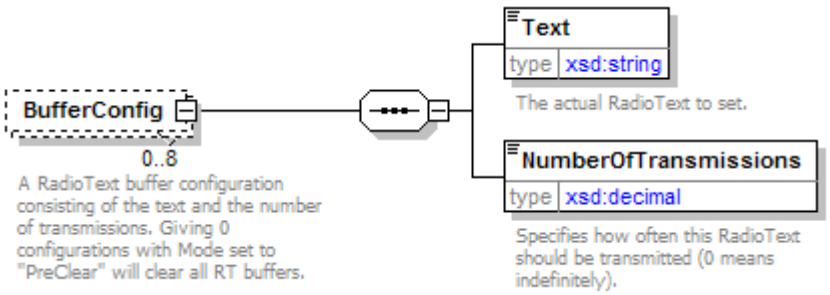
diagram	
properties	isRef 0 content complex
children	**BufferConfig**

attributes

Name	Type	Use	Default	Fixed	Annotation
Mode	**derived by:** **xsd:string**	optional	PreClear		documentation Specifies, if all RT buffers should be cleared before appending the new RadioText or if it should get appended to the existing RT buffers.

annotation	documentation RadioText

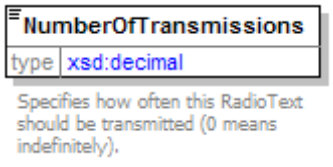
element UECPMessage/Message/Command/RT/BufferConfig

diagram	
properties	isRef 0 minOcc 0 maxOcc 8 content complex
children	**Text, NumberOfTransmissions**
annotation	documentation A RadioText buffer configuration consisting of the text and the number of transmissions. Giving 0 configurations with Mode set to "PreClear" will clear all RT buffers.

element UECPMessage/Message/Command/RT/BufferConfig/Text

diagram	
type	restriction of **xsd:string**
properties	isRef 0 content simple
facets	maxLength 64
annotation	documentation The actual RadioText to set.

element UECPMessage/Message/Command/RT/BufferConfig/NumberOfTransmissions

diagram	
type	restriction of **xsd:decimal**
properties	isRef 0 content simple
facets	minInclusive 0 maxInclusive 15
annotation	documentation Specifies how often this RadioText should be transmitted (0 means indefinitely).

element UECPMessage/Message/Command/FreeUECP

diagram	**FreeUECP** type xsd:string A free UECP command given as comma separated hexadecimal values. The CRC generation and sequence counter as well as the start / stop sequence and length calculation of the UECP command will be inserted automatically. Example: "1E,01" to switch RDS output signal "On".
type	restriction of **xsd:string**
properties	isRef 0 content simple
facets	minLength 2 maxLength 764 pattern [0-9A-Fa-f]{2}(,[0-9A-Fa-f]{2})*
annotation	documentation A free UECP command given as comma separated hexadecimal values. The CRC generation and sequence counter as well as the start / stop sequence and length calculation of the UECP command will be inserted automatically. Example: "1E,01" to switch RDS output signal "On".

6.3 EC Exchange Format

This specification describes the interchange format between data generated by the OCTO IP server and the Encoder-Check (optional). The Encoder-Check only reads the content and uses it for addressing the encoders. The list is a comma-separated list in ASCII. Comments are written with a leading number sign (#) in the first position for a single line.

File:

<*UECP-Address*>,[<IP-Address>],[<Symbolic-Name>],<*PI-Number*>,<*Spare-Encoder*>

Where:

<*UECP-Address*> is a 16bit hexadecimal number and represents the UECP-Address
<IP-Address> is the optional IP-Address of the encoder for error reporting purposes only
<Symbolic-Name> is the optional name of the encoder for error reporting purposes only
<*PI-Number*> is a 16bit hexadecimal number and represents the Program-Identification
<*Spare-Encoder*> is 1 for spare, and 0 for a regular encoder

Example:

```
 1,,,1,0
 2,,,2,0
 5,192.168.237.5,,3,0
#
# spare encoder is for test purposes only, comment this out if not active
#
#3,,,4,1
3,,,4,0
01C3,,city_station1,ABCD,1
```

6.4 Log files

Several different log files can be generated when operating the OCTO IP server software, e.g. for Message Queues, GPI/Os or socket connections. These event logs can be activated for the inputs and the outputs. Error logs are always activated. All the generated log and error files will be written the way *open, append, close*. The files do have the following format and content: (Note: Variable values are *ITALIC).*

6.4.1 Input Log

*.log

File Type	*.log	directory 'log\input\'
File name format	'*dd.mm.yyyy_hh_type_port*.log'	'*dd.mm.yyyy*" is 'day/month/year', 'hh' is full hour, '*type'* is e.g. UECP, '*port'* is the IP or COM port number
Description	Logs the data frames of an input when enabled in the configuration.	
Possible Messages	Some MECs (e.g. TA/PTY/REQ) are placed in readable form, others in hex. At the first position a data direction symbol and a time stamp is placed. "<" means incoming data, ">" means outgoing data.	
Example <14:00:01 REQ 192.1.1.1 'Coder1' FE,00,41,5A,05,17,03,01,00,00,C3,34,FF		
Description Example	At 14:00:01 a REQ was received from IP address 192.1.1.1, configured with the destination label (or group) Coder1 and hex data.	

*.err

File Type	*.err	directory 'log\input\'
File name format	'*dd.mm.yyyy_hh_type_port*.err.log'	'*dd.mm.yyyy*" is 'day/month/year', 'hh' is full hour, '*type'* is e.g. UECP, '*port'* is the IP or COM port number
Description	Logs the error frames (no option, always enabled)	
Possible Messages	e.g. 'No Target with UECP-address'	
Example 14:00:02 REQ - 'No Target with UECP-address:0048' FE,00,48,60,05,17,03,01,00,00,30,FD,01,FF		
Description Example	At 14 :00 :02 a REQ was sent, but the appropriate encoder with UECP address 0048 was not available, the sent data is following.	

6.4.2 Output Log

*.log

File Type	*.log	directory ‘log\output\’
File name format	‘*dd.mm.yyyy_hh_dest*.log’	‘*dd.mm.yyyy*” is ‘day/month/year’, ‘hh’ is full hour, ‘*type’* is e.g. UECP, ‘*dest’* is the destination IP address or COM port number. The omitted data type is always UECP.
Description	Logs the outgoing data frames of an output when enabled in the configuration.	
Possible Messages	Some MECs (e.g. TA/PTY/REQ) are placed in readable form, others in hex. At the first position a data direction symbol and a time stamp is placed. “<” means incoming data, “>” means outgoing data.	
Example >12:00:00 PI UECP ‘6667’ FE,00,00,18,05,01,00,0A,3E,E8,37,76,FF		
Description Example	At 12:00:00 the data type PI was received from source port 6667 for forwarding to the destination (see log file name). The corresponding data is following.	

*.err

File Type	*.err	directory ‘log\output\’
File name format	‘*dd.mm.yyyy_hh _dest*.err.log’	‘*dd.mm.yyyy*” is ‘day/month/year’, ‘hh’ is full hour, ‘*dest’* is the destination IP address or COM port number.
Description	Logs the error frames (no option, always enabled).	
Possible Messages	e.g. ‘Data connection lost’	
Example 12:00:42 Connection lost !		
Description Example	At 12:00:42 the data connection was lost.	

6.5 OCTO_IP.ini

OCTO_IP uses the configuration file “OCTO_IP.ini”, for several settings. The file has the path “\config\OCTO_IP.ini” in the installation folder. This file can be edited by the user with care and contains the following sections:

XML Output	Enter here the paths to your own specific XML definitions.
Groups	Here you can change the default names for the routing groups to your own definitions. For group addressing you can use short group names that represent the RDS encoders and the logging outputs. For PSN addressing you need to use special group names to correspond to the PSN of your RDS encoders. The format is: PSN_0*hexvalue* (e.g. PSN_0A). Use upper case characters for the hex values.
WebIO	Enter here the IP addresses of all possible 2wcom GP I/O boxes. Only then you are able to use them with OCTO-IP.

7 Support

More often than not, it is only a small detail that has been overlooked and leads to a problem. Therefore, please read the user manual carefully, as this will help you to understand, prevent and eliminate typical problems. Please report failures and problems by email to support@2wcom.com.