ONELAN Digital Signage:

Digital Signage Manager Technical Reference

ONELAN Digital Signage

V3.1.0 (30834)

Digital Signage Manager
Technical Reference
V3.1.0

by ONELAN Limited

The ONELAN Digital Signage Manager is a Web appliance designed to monitor and remotely manage multiple ONELAN digital signage Net-Top-Boxes (NTBs). It allows remote configuration, provides monitoring, maintenance, and media playout audit reports on groups of NTBs over user-defined periods.

Printed: May 2012

ONELAN Digital Signage

Table of Contents

ONELAN Digital Signage

Part 1

Introduction

1 Introduction

This document describes the application programming interfaces (APIs) to the Digital Signage Manager (DSM).

The three API types are:

- XML API (read and write access).
- JSON and JSONP API (read-only access).
- Database views (read-only access).

Within this document you will find:

- Detailed information about each API.
- The user permissions required to execute each API function.
- Code examples.

1.1 Target Audience

This technical reference is intended for system integrators who want to integrate DSM into an enterprise environment. It is assumed you are familiar with the DSM user interface and with the contents of the *Digital Signage Manager User Guide.* You require a good understanding of web technologies to use the APIs.

1.2 API Features

The API to use will depend on the features you wish to implement. This document is divided into parts, one for each type:

- **XML API** 7 – Configuring and managing Organisations, users, players, and alarms on a DSM.
- **JSON API** 32 – Obtaining information from the DSM about players and Organisations.
- **Database Views** 39 – Accessing the information held in the DSM database views.

ONELAN Digital Signage

Part II

XML API

2 XML API

The XML API provides a way to configure and manage Organisations, users, players and alarms on a DSM.

To submit a request to the DSM, you must observe these requirements:

- Set the HTTP header field `Content-type` to `text/xml`.
- Wrap all of your XML API commands in a container called `<command_list>` (see Command List 7 for details).
- Use HTTP or HTTPS to post your commands to /XML1 with the credentials of a user with the required permissions (see Command List 7 for details of permissions).

DSM returns a response to the request in a container called `<response_list>` (see Response List 8 for details).

2.1 Command List

When composing your XML API code, be aware of these characteristics of `<command_list>`:

- It can contain multiple commands.
- It cannot contain a mixture of read-only and state-changing commands.
- Processing stops on the first error unless you specify `<command_list continue_on_error="yes">`.

Before relying on a particular command, you may want to confirm that the target version of the DSM supports it. Use the `get_all_capabilities` command to do so (see Checking Capabilities 9 for details).

Syntax

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<command_list>
    <command_required command_id="xx"/>
    <command_required command_id="xx"/>
    .
    .
    .
</command_list>
```

XML Node	Notes
`<command_list>`	Root element for all requests.
`<command_required>`	Command name.
`command_id`	A string you enter to uniquely identify each command. DSM tags each response in the `<response_list>` with the same `command_id`.

Following is the list of supported commands and the user permissions required to execute them:

Command Name	User Permission Required	Purpose and Further Details
`get_all_capabilities`	Any of the API permissions	Checking Capabilities 9
`organisation_add`	API System Administration	Adding an Organisation 11

ONELAN Digital Signage

Command Name	User Permission Required	Purpose and Further Details
organisation_delete	API System Administration	Deleting an Organisation 13
organisation_get_settings	API System Administration	Getting Organisation Settings 14
organisation_set_settings	API System Administration	Updating Organisation Settings 15
user_add	API System Administration	Adding a User 18
user_delete	API System Administration	Deleting a User 19
user_get_settings	API System Administration	Getting User Settings 20
user_set_settings	API System Administration	Updating User Settings 21
player_add	API System Administration	Adding a Player 23
player_delete	API System Administration	Deleting a Player 24
player_get_settings	API System Administration	Getting Player Settings 25
player_set_settings	API System Administration	Updating Player Settings 26
player_set_name	API Organisation Administration	Setting Player Name 27
player_get_alarm_info	API Organisation Status View	Getting Player Alarm Information 28
player_alarm_ack	API Organisation Alarm Update	Acknowledging an Alarm 29
player_alarm_unack	API Organisation Alarm Update	Unacknowledging an Alarm 30

2.2 Response List

DSM returns a response in a container called <response_list>. DSM tags each response in the <response_list> with the same command identifier you entered for the matching command in the request <command_list>.

The status_code in the <response_list> element reflects the overall result of all the commands in the <command_list> . If all commands executed successfully, DSM sets the status_code to 0.

Syntax

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="x">
    <command_requested command_id="xx" status_code="x">
       <status>xxxxxxxxxxxxxxxx</status>
    </command_requested>
</response_list>
```

ONELAN Digital Signage

XML Node	Notes
<response_list>	Container for all responses.
status_code	Indicates the overall return status for the <command_list>: 0 All commands executed successfully. 1 Invalid XML. 2 Partial failure: at least one command failed. 3 <command_list> contained a mix of read-only and state-changing commands. 4 The request was not sent with the correct HTTP headers. 99For all other errors.
<*command_requested*>	Command for which DSM returns this response.
command_id	The unique command identifier you specified in the <command_list>.
status_code	Indicates the status return for the individual command in the <command_list > (each command group also has specific codes starting from 100): 0 DSM executed the command successfully. 1 There was an error whilst trying to execute the command. 2 DSM did not execute the command as a previous command in the <command_list> failed to execute. 3 The user did not have the correct permission to execute the command.
<status>	Text describing the command's success or failure.

2.3 Checking Capabilities

The XML API provides a capability-checking mechanism for you to determine if the target version of DSM supports the commands you want to use.

This mechanism can help enhance your application's robustness. By first checking for the supported commands, you can implement an alternative approach if the command you want is not available.

Note: For future viability, this is an assured backwards-compatible interface.

Syntax

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<command_list>
    <get_all_capabilities command_id="xx"/>
</command_list>
```

XML Node	Notes
<command_list>	Root element for all requests.
<get_all_capabilities>	Command to check for capabilities available on the DSM.
command_id	Uniquely identifies the command instance.

Permission Required

Any of the API permissions.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<command_list>
    <get_all_capabilities command_id="get cmd caps id"/>
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
    <get_all_capabilities command_id="get cmd caps id" status_code="0">
       <capability name="CAP_USER_ADD_1"/>
       <capability name="CAP_USER_DEL_1"/>
       .
       .
       .
    </get_all_capabilities>
</response_list>
```

Following are the capability names returned and the commands they indicate the DSM supports:

Capability Name Returned	Command Supported
CAP_ORG_ADD_1	organisation_add
CAP_ORG_DEL_1	organisation_delete
CAP_ORG_GET_SETTINGS_1	organisation_get_settings
CAP_ORG_SET_SETTINGS_1	organisation_set_settings
CAP_USER_ADD_1	user_add
CAP_USER_DEL_1	user_delete
CAP_USER_GET_SETTINGS_1	user_get_settings
CAP_USER_SET_SETTINGS_1	user_set_settings
CAP_PLAYER_ADD_1	player_add

Capability Name Returned	Command Supported
CAP_PLAYER_DEL_1	player_delete
CAP_PLAYER_GET_SETTINGS_1	player_get_settings
CAP_PLAYER_SET_SETTINGS_1	player_set_settings
CAP_PLAYER_SET_NAME_1	player_set_name
CAP_PLAYER_GET_ALARM_INFO_1	player_get_alarm_info
CAP_PLAYER_ALARM_ACK_1	player_alarm_ack
CAP_PLAYER_ALARM_UNACK_1	player_alarm_unack

2.4 Organisation Management

Use the Organisation management group of commands to:

- Add an Organisation 11
- Delete an Organisation 13
- Get an Organisation's settings 14
- Update an Organisation's settings 15

Following are the return status codes specific to this group of commands:

Status Code	Meaning
100	The Organisation already exists.
101	The Organisation does not exist.
102	The Organisation cannot be deleted as it still has users.
103	The Organisation cannot be deleted as it still has players.

2.4.1 Adding an Organisation

Use the `organisation_add` command to add a new Organisation.

Syntax

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <organisation_add command_id="xx" name="xx">
        <min_reporting_interval value="xx"/>
        <allow_alarm_ack value="xx"/>
        <allow_alarm_config value="xx"/>
        <allow_alarm_email value="xx"/>
        <allow_rs232_reporting value="xx"/>
        <allow_thumbnail_reporting value="xx"/>
        <max_players value="xx"/>
```

```
        <reports_retention_period value="xx"/>
        <mae_retention_period value="xx"/>
    </organisation_add>
</command_list>
```

XML Node	Notes	Default (If Value Not Specified)
<organisation_add>	Command to add a new Organisation.	
command_id	Uniquely identifies the command instance.	
name	The Organisation name.	
<min_reporting_interval>	The minimum reporting interval in seconds.	
value	The number of seconds.	60
<allow_alarm_ack>	Enable alarm acknowledgement for this Organisation.	
value	Either yes or no.	yes
<allow_alarm_config>	Enable alarm settings to be configured for this Organisation.	
value	Either yes or no.	yes
<allow_alarm_email>	Enable alarms to be emailed for this Organisation.	
value	Either yes or no.	yes
<allow_rs232_reporting>	Enable RS-232 reports for this Organisation.	
value	Either yes or no.	yes
<allow_thumbnail_reporting>	Enable thumbnail reports for this Organisation.	
value	Either yes or no.	yes
<max_players>	The total number of players that can be added to the Organisation.	
value	The number of players.	Maximum allowed by license.
<reports_retention_period>	If you enable **Prune Database** on the DSM, it removes report data older than this period.	
value	The length of the retention period in days.	90
<mae_retention_period>	If you enable **Prune Database** on the DSM, it removes media audit events older than this period.	
value	The length of the retention period in days.	90

Permission Required

API System Administration

Example

Request

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <organisation_add  name="org1" command_id="user specified id">
       <min_reporting_interval value="15" />
       <allow_alarm_ack value="yes"/>
       <allow_alarm_email value="yes"/>
       <allow_rs232_reporting value="yes"/>
       <allow_thumbnail_reporting value="yes"/>
       <max_players value="100" />
       <reports_retention_period value="100"/>
       <mae_retention_period value="100"/>
    </organisation_add>
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
    <organisation_add status_code="0" command_id="user specified id">
        <status>Added Organisation org1</status>
    </organisation_add>
</response_list>
```

2.4.2 Deleting an Organisation

Use the `organisation_delete` command to delete an Organisation.

Syntax

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <organisation_delete command_id="xx" name="xx"/>
</command_list>
```

XML Node	Notes
`<organisation_delete>`	Command to delete an Organisation. The Organisation must not have any users or players for this to succeed.
`command_id`	Uniquely identifies the command instance.
`name`	The Organisation name.

Permission Required

API System Administration

Example

Request

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <organisation_delete  name="org1" command_id="delete organisation 1"/>
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
    <organisation_delete status_code="0" command_id="delete organisation 1">
        <status>Deleted Organisation org1</status>
    </organisation_delete>
</response_list>
```

2.4.3 Getting Organisation Settings

Use the `organisation_get_settings` command to check an Organisation's settings and permissions. The syntax of the `<response_list>` parallels that of the `<command_list>` described in Adding an Organisation 11.

Syntax

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <organisation_get_settings command_id="xx" name="xx"/>
</command_list>
```

XML Node	Notes
`<organisation_get_settings>`	Command to get an Organisation settings.
`command_id`	Uniquely identifies the command instance.
`name`	The Organisation name.

Permission Required

API System Administration

Example

Request

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <organisation_get_settings  name="org1" command_id="get org1 settings"/>
</command_list>
```

Response

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<response_list status_code="0">
    <organisation_get_settings name="org1" command_id="get org1 settings" status_code="0">
```

```
        <min_reporting_interval value="15" />
        <allow_alarm_acknowledgement value="yes"/>
        <allow_alarm_config value="yes"/>
        <allow_alarm_email value="yes"/>
        <allow_rs232_reporting value="yes"/>
        <allow_thumbnail_reporting value="yes"/>
        <max_players value="100" />
        <reports_retention_period value="100"/>
        <mae_retention_period value="100"/>
    </organisation_get_settings>
</response_list>
```

2.4.4 Updating Organisation Settings

Use the `organisation_set_settings` command to update an Organisation's settings and permissions.

Syntax

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<command_list>
    <organisation_set_settings command_id="xx" name="xx" >
        <min_reporting_interval value="xx"/>
        <allow_alarm_ack value="xx"/>
        <allow_alarm_config value="xx"/>
        <allow_alarm_email value="xx"/>
        <allow_rs232_reporting value="xx"/>
        <allow_thumbnail_reporting value="xx"/>
        <max_players value="xx"/>
        <reports_retention_period value="xx"/>
        <mae_retention_period value="xx"/>
    </organisation_set_settings>
</command_list>
```

XML Node	Notes	Default
`<organisation_set_settings>`	Command to update an Organisation settings.	
`command_id`	Uniquely identifies the command instance.	
`name`	The Organisation name.	
`<min_reporting_interval>`	The minimum reporting interval in seconds.	Each of these settings: • Retains its previous value if you do not include the relevant XML node. • Resets to its default value if you include the relevant XML node but without the `value` attribute. See Adding an Organisation 11 for the default values.
`value`	The number of seconds.	
`<allow_alarm_ack>`	Enable alarm acknowledgement for this Organisation.	
`value`	Either `yes` or `no`.	
`<allow_alarm_config>`	Enable alarm settings to be configured for this Organisation.	
`value`	Either `yes` or `no`.	

XML Node	Notes	Default
<allow_alarm_email>	Enable alarms to be emailed for this Organisation.	
value	Either yes or no.	
<allow_rs232_reporting>	Enable RS-232 reports for this Organisation.	
value	Either yes or no.	
<allow_thumbnail_reporting>	Enable thumbnail reports for this Organisation.	
value	Either yes or no.	
<max_players>	The total number of players that can be added to the Organisation.	
value	The number of players.	
<reports_retention_period>	If you enable **Prune Database** on the DSM, it removes report data older than this period.	
value	The length of the retention period in days.	
<mae_retention_period>	If you enable **Prune Database** on the DSM, it removes media audit events older than this period.	
value	The length of the retention period in days.	

Permission Required

API System Administration

Example

Request

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <organisation_set_settings name="org1" command_id="set org1 settings">
       <min_reporting_interval value="15" />
       <allow_alarm_ack value="yes"/>
       <allow_alarm_email value="yes"/>
       <allow_rs232_reporting value="no"/>
       <allow_thumbnail_reporting value="no"/>
       <max_players value="100" />
       <reports_retention_period value="100"/>
       <mae_retention_period value="100"/>
    </organisation_set_settings>
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
```

```
    <organisation_set_settings status_code="0" command_id="set org1 settings">
        <status>Updated settings for organisation org1</status>
    </organisation_set_settings>
</response_list>
```

2.5 User Management

Use the user management group of commands to:

- Add a user 18
- Delete a user 19
- Get a user's settings 20
- Update a user's settings 21

You set permissions to control what users can and cannot do on the DSM.The following table lists the web UI permission names and the corresponding values for assigning them using the XML API:

Web UI Permission	XML API Value
Organisation Status View	org_status_view
Organisation Alarm Update	org_alarm_update
Organisation Administration	org_admin
Organisation Browse to Player via VPN	org_vpn_http
Organisation SSH to Player via VPN	org_vpn_ssh
Organisation Media Audit View	org_media_audit_view
Accept Report	accept_report
Establish VPN	vpn_allowed
External Database Reports	ext_dbview_basic
FTP Access	ftp_access
System Administration	sysadmin
Database Backup	db_backup
API System Administration	api_sysadmin
API Organisation Administration	api_org_admin
API Organisation Alarm Update	api_org_alarm_update
API Organisation Status View	api_org_status_view

Following are the return status codes specific to this group of commands:

Status Code	Meaning
100	The Organisation does not exist.
101	The user does not exist.
102	The user must have at least one Organisation permission.

Status Code	Meaning
103	The permissions assigned to the user are incompatible.
104	You cannot remove **System Administration** permission from the only system administrator user.
105	The user already exists.
106	The user is not a member of the Organisation for unknown players.
107	You cannot delete the current user.

2.5.1 Adding a User

Use the `user_add` command to add a new user.

Syntax

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<command_list>
    <user_add command_id="xx" name="xx" password="xx">
        <member_of>
            <organisation name="xx"/>
        </member_of>
        <add_unknown_players_to_organisation name="xx"/>
        <permissions>
            <permission name="xx"/>
        </permissions>
    </user_add>
</command_list>
```

XML Node	Notes
`<user_add>`	Command to add a new user.
`command_id`	Uniquely identifies the command instance.
`name`	User's username.
`password`	User's password (in clear text).
`<member_of>`	Container holding the Organisations to which to add the user.
`<organisation>`	One node for each Organisation.
`name`	Name of the Organisation.
`<add_unknown_players_to_organisation>`	Enables the user to add unknown players to the Organisation.
`name`	Name of the Organisation.
`<permissions>`	Container for assigning permissions to the user.
`<permission>`	One node for each permission.
`name`	Permission name (listed in User Management 17).

Permission Required

API System Administration

Example

Request

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
  <user_add command_id="new user add" name="reporting_user" password=”xyZ12Se”>
   <member_of>
       <organisation name="org1"/>
       <organisation name="org2"/>
   </member_of>
   <add_unknown_players_to_organisation name="org1"/>
   <permissions>
       <permission name="accept_report"/>
       <permission name="vpn_allowed"/>
   </permissions>
  </user_add>
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
    <user_add status_code="0" command_id="new user add">
        <status>Added user reporting_user</status>
    </user_add>
</response_list>
```

2.5.2 Deleting a User

Use the `user_delete` command to delete a user.

Syntax

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
     <user_delete command_id="xx" name="xx"/>
</command_list>
```

XML Node	Notes
`<user_delete>`	Command to delete an existing user.
`command_id`	Uniquely identifies the command instance.
`name`	User’s username.

Permission Required

API System Administration

Example

Request

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
     <user_delete command_id="delete user" name="reporting_user"/>
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
    <user_delete status_code="0" command_id="delete user">
        <status>Deleted user reporting_user</status>
    </user_delete>
</response_list>
```

2.5.3 Getting User Settings

Use the `user_get_settings` command to check the user's Organisation and permissions. The syntax of the `<response_list>` parallels that of the `<command_list>` described in Adding a User 18.

Syntax

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <user_get_settings command_id="xx" name="xx"/>
</command_list>
```

XML Node	Notes
`<user_get_settings>`	Command to get a user’s settings.
`command_id`	Uniquely identifies the command instance.
`name`	User’s username.

Permission Required

API System Administration

Example

Request

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <user_get_settings command_id="get user settings" name="reporting_user"/>
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
    <user_get_settings status_code="0" command_id="get user settings">
```

```
        <member_of>
            <organisation name="org1"/>
            <organisation name="org2"/>
        </member_of>
        <permissions>
            <permission name="accept_report"/>
            <permission name="vpn_allowed"/>
        </permissions>
        <add_unknown_players_to_organisation name=org1/>
    </user_get_settings>
</response_list>
```

2.5.4 Updating User Settings

Use the `user_set_settings` command to update the user's Organisation and permissions.

Syntax

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<command_list>
    < user_set_settings command_id="xx" name="xx" password="xx">
        <member_of>
            <organisation name="xx"/>
        </member_of>
        <add_unknown_players_to_organisation name="xx"/>
        <permissions>
            <permission name="xx"/>
        </permissions>
    </user_set_settings>
</command_list>
```

XML Node	Notes	Default
`<user_set_settings>`	Command to update a user's settings.	
`command_id`	Uniquely identifies the command instance.	
`name`	Name of user to update.	
`password`	New password.	If missing, the user retains the existing password.
`<member_of>`	Container holding the Organisations to which to add the user.	The outcome depends on how you implement the XML node: • If missing, the user's Organisation remains the same. • If you Include an empty `<member_of>`, it removes the user from all Organisations.
`<organisation>`	One node for each Organisation.	
`name`	Name of the Organisation.	

XML Node	Notes	Default
		• If you name an Organisation, it replaces the user's current Organisation with the named one.
<add_unknown_players_to_organisation>	Enables the user to add unknown players to the Organisation.	The outcome depends on how you implement the XML node: • If missing, the user retains the existing setting. • If present without the name attribute, the user will ignore unknown players. • If present with the name attribute, the user will add unknown players to the named Organisation.
name	Name of the Organisation.	
<permissions>	Container for assigning permissions to the user.	The outcome depends on how you implement the XML node: • If missing, the user retains the existing permissions. • If included, replaces the user's permissions with the new set.
<permission>	One node for each permission.	
name	Permission name (listed in User Management 17).	

Permission Required

API System Administration

Example

Request

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    < user_set_settings command_id="update user" name="reporting_user">
        <member_of>
            <organisation name="org1"/>
        </member_of>
        <add_unknown_players_to_organisation name="org1"/>
    </user_set_settings>
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
    <user_set_settings status_code="0" command_id="update user">
        <status>Updated user reporting_user</status>
    </user_set_settings>
</response_list>
```

2.6 Player Management

Use the player management group of commands to:

- Add a player 23
- Delete a player 24
- Get a player's settings 25
- Update a player's settings 26
- Set a player's name 27

Following are the return status codes specific to this group of commands:

Status Code	Meaning
100	Unknown Organisation.
101	The player already exists.
102	Another player with same name already exists.
103	You cannot add more players as the license limit has been reached.
104	You cannot add more players as the Organisation's maximum players has been reached.
105	The player does not exist.

2.6.1 Adding a Player

Use the `player_add` command to add a new player.

Syntax

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<command_list>
     <player_add command_id="xx" serial="xx" name="xx" organisation="xx"  accept_reports="xx"/>
</command_list>
```

XML Node	Notes
`<player_add>`	Command to add a new player.
`command_id`	Uniquely identifies the command instance.
`serial`	Player serial number.
`organisation`	The Organisation to which to add the player.
`name`	Optional. A meaningful name for the player.
`accept_reports`	If `yes`, the Organisation will accept reports from the player. If `no`, the Organisation rejects reports from the player.

Permission Required

API System Administration

Example

Request

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
     <player_add command_id="add player" serial="14533"
      organisation="org1" name="reception area player" accept_reports="yes"/>
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
    <player_add status_code="0" command_id="add player">
        <status>Player 14533 added</status>
    </player_add>
</response_list>
```

2.6.2 Deleting a Player

Use the `player_delete` command to delete a player.

Syntax

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <player_delete command_id="xx" serial="xx"/>
</command_list>
```

XML Node	Notes
`<player_delete>`	Command to delete an existing player. This also deletes all reports associated with the player.
`command_id`	Uniquely identifies the command instance.
`serial`	Player serial number.

Permission Required

API System Administration

Example

Request

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <player_delete command_id="delete player" serial="14533"/>
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
    <player_delete status_code="0" command_id="delete player">
        <status>Deleted Player 14533</status>
    </player_delete>
</response_list>
```

2.6.3 Getting Player Settings

Use the `player_get_settings` command to check a player's name, Organisation, and whether it accepts reports. The syntax of the `<response_list>` parallels that of the `<command_list>` described in Adding a Player 23.

Syntax

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <player_get_settings command_id="xx" serial="xx"/>
</command_list>
```

XML Node	Notes
`<player_get_settings>`	Command to get a player’s settings.
`command_id`	Uniquely identifies the command instance.
`serial`	Player serial number.

Permission Required

API System Administration

Example

Request

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <player_get_settings command_id="get player settings" serial="14533"/>
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
    <player_get_settings status_code="0" command_id="get player settings"
     name="player reception area" accept_reports="yes" organisation="org1">
    </player_get_settings>
</response_list>
```

2.6.4 Updating Player Settings

Use the `player_set_settings` command to update a player's name, Organisation, and change whether the Organisation accepts its reports.

Syntax

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<command_list>
    <player_set_settings command_id="xx" serial="xx" name="xx" organisation="xx" accept_reports="xx"/>
</command_list>
```

XML Node	Notes	Default
`<player_set_settings>`	Command to update a player's settings.	
`command_id`	Uniquely identifies the command instance.	
`serial`	Player serial number.	
`organisation`	The Organisation to which to add the player.	The outcome depends on what you do with the parameter: • Do not include it – the player retains the existing setting. • Include it – the player's setting changes accordingly.
`name`	Optional. A meaningful name for the player.	
`accept_reports`	If `yes`, the Organisation will accept reports from the player. If `no`, the Organisation rejects reports from the player.	

Permission Required

API System Administration

Example

Request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<command_list>
     <player_set_settings command_id="set player settings" serial="14533" name="new office"/>
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
    <player_set_settings status_code="0" command_id="set player settings">
        <status>Player 14533 updated</status>
    </player_set_settings>
</response_list>
```

2.6.5 Setting Player Name

Use the `player_set_name` command to change a player's name.

Syntax

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <player_set_name command_id="xx" serial="xx" name="xx"/>
</command_list>
```

XML Node	Notes
`<player_set_name>`	Command to update a player’s name.
`command_id`	Uniquely identifies the command instance.
`serial`	Player serial number.
`name`	Player new name.

Permission Required

API Organisation Administration

Example

Request

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <player_set_name command_id="update player name" serial="14533" name="old office"/>
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
    <player_set_name status_code="0" command_id="update player name">
        <status>Player name set to old office.</status>
    </player_set_name>
</response_list>
```

2.7 Alarm Management

Use the alarm management group of commands to:

- Get player alarm information 28
- Acknowledge an alarm 29
- Unacknowledge an alarm 30

Following are the return status codes specific to this group of commands:

Status Code	Meaning
100	The alarm ID does not exist.
101	Alarm acknowledgement is disabled for this Organisation.
102	The alarm has already been lowered.

2.7.1 Getting Player Alarm Information

Use the `player_get_alarm_info` command to check the alarms raised for a player. The `<response_list>` includes the `<alarm_info>` element containing:

- `name` – The name of the alarm (see the *Digital Signage Manager User Guide* for details)
- `id` – The unique alarm identifier generated by the DSM.
- `raise_time` – Timestamp when the DSM raised the alarm.
- `acked` – Whether a user has acknowledged the alarm.

Syntax

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <player_get_alarm_info command_id="xx" serial="xx"/>
</command_list>
```

XML Node	Notes
`<player_get_alarm_info>`	Command to get alarm information about a player.
`command_id`	Uniquely identifies the command instance.
`serial`	Player serial number.

Permission Required

API Organisation Status View

Example

Request

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
```

```
<command_list>
    <player_get_alarm_info command_id="get player alarms" serial="14533" />
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
    <player_get_alarm_info status_code="0" command_id="get player alarms">
        <alarm_info
         name="Subscriber Error" id="1" raise_time="2011-10-10 15:16:05" acked="no"/>
        <alarm_info
         name="Subscriber Has No Publisher" id="2" raise_time="2011-10-10 15:16:05" acked="no"/>
    </player_get_alarm_info>
</response_list>
```

2.7.2 Acknowledging an Alarm

Use the `player_alarm_ack` command to acknowledge an alarm.

Syntax

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <player_alarm_ack command_id="xx" id="xx" reason="xx"/>
</command_list>
```

XML Node	Notes
`<player_alarm_ack>`	Command to acknowledge an alarm instance.
`command_id`	Uniquely identifies the command instance.
`id`	The alarm ID returned in the response to the relevant `player_get_alarm_info`.
`reason`	Description of your response to the alarm (the reason for acknowledging it).

Permission Required

API Organisation Alarm Update

Example

Request

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <player_alarm_ack command_id="ack alarm" id="1" reason="Publish under maintenance"/>
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
    <player_alarm_ack status_code="0" command_id="ack alarms">
        <status>Acknowledged alarm id 1</status>
```

```
    </player_alarm_ack>
</response_list>
```

2.7.3 Unacknowledging an Alarm

Use the `player_alarm_unack` command to unacknowledge an alarm.

Syntax

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <player_alarm_unack command_id="xx" id="xx"/>
</command_list>
```

XML Node	Notes
`<player_alarm_unack>`	Command to unacknowledge an alarm instance.
`command_id`	Uniquely identifies the command instance.
`id`	The alarm ID returned in the response to the relevant `player_get_alarm_info`.

Permission Required

API Organisation Alarm Update

Example

Request

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<command_list>
    <player_alarm_unack command_id="unack alarm" id="1"/>
</command_list>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response_list status_code="0">
    <player_alarm_unack status_code="0" command_id="unack alarm">
        <status>Unacknowledged alarm id 1</status>
    </player_alarm_unack>
</response_list>
```

Part III

JSON and JSONP API

3 JSON and JSONP API

The JSON API provides a read-only mechanism for obtaining data about players and Organisations from the DSM. JSONP allows cross-domain sharing of data.

Use the JSON API resources to:

- List all players 32
- Get player status 33
- List all Organisations 37

The following query parameters apply to all of the JSON API:

Parameter	Value	Effect
`indent`	Any positive integer value (for example, 2)	The output is pretty formatted.
`callback`	JavaScript function name	Turns returned value into JSONP.

3.1 Listing all Players

URI

`/JSON1/players/all`

Permission Required

API System Administration

Examples

URI

`/JSON1/players/all`

Response

```
{"all_players": [{"player_serial": 29926, "report_id": 181}]}
```

URI

`/JSON1/players/all?indent=2`

Response

```
{
  "all_players": [
    {
      "player_serial": 29926,
      "report_id": 181
    }
  ]
}
```

URI

```
/JSON1/players/all?indent=2&callback=processAllPlayers
```

Response

```
processAllPlayers({
  "all_players": [
    {
      "player_serial": 29926,
      "report_id": 181
    }
  ]
})
```

URI

```
/JSON1/players/all?callback=processAllPlayers
```

Response

```
processAllPlayers({"all_players": [{"player_serial": 29926, "report_id": 181}]})
```

3.2 Getting Player Status

URI

`/JSON1/player/serial/status`

Complete the query by including the player serial number in the *`serial`* part of the URI.

Optionally, to specify the exact information you require, modify the request using the `fields` parameter as follows:

Parameter	Value	Effect
`fields`	Comma separated list of values with no spaces. You may include any of: • `alarms` • `channel` • `system` • `local_info` • `rs232` • `remote_setup` • `schedule`	The response from the DSM is: • Pretty formatted by suitable indents for each level relative to the parent. • Expanded into detailed status information for each value you specify.

Permission Required

API Organisation Status View

Examples

Full Player Status Information

URI

```
/JSON1/player/14533/status?indent=2
```

Response


```
{
  "serial_number": 14533,
  "player_name": null,
  "reported_at": "2011-10-11T09:59:05+01:00",
  "alarms": [
    {
      "alarm_name": "schedule_not_normal",
      "alarm_info": {
        "is_observed": true,
        "acknowledge": {},
        "raise_time": "2011-10-10T18:01:05+01:00",
        "lower_time": null
      }
    },
    {
      "alarm_name": "subscriber_error",
      "alarm_info": {
        "is_observed": true,
        "acknowledge": {},
        "raise_time": "2011-10-10T15:16:05+01:00",
        "lower_time": null
      }
    },
    {
      "alarm_name": "subscriber_no_publisher",
      "alarm_info": {
        "is_observed": true,
        "acknowledge": {},
        "raise_time": "2011-10-10T15:16:05+01:00",
        "lower_time": null
      }
    }
  ],
  "channel_role": "subscriber",
  "all_channels": [
    {
      "type": "ftp",
      "name": null,
      "channel_id": null,
      "url": "ftp://testproxy/1234",
      "channel_number": "1",
      "active": true,
      "status": {
        "files": {
          "transferred": null,
          "total": null
        },
```

```
        "last_error": "Error while Publishing: Problems with Layouts or media prevent publishing",
        "size": {
          "transferred": null,
          "total": null
        }
      }
    }
  ],
  "system": {
    "up_since": "2011-10-06T15:23:53+01:00",
    "hardware": {
      "model": "NTB655",
      "oem_model": "NTB655P",
      "disk_temperature": 34.0,
      "disk_safety_margin": 21.0,
      "cpu_temperature": 50.0,
      "cpu_safety_margin": 45.0,
      "cpu_fan_speed": null,
      "system_fan_speed": 2481,
      "system_temperature": 41.0,
      "data_disk": {
        "free": 145683062784,
        "total": 146841247744
      },
      "system_disk": {
        "free": 2067959808,
        "total": 4227530752
      }
    },
    "software": {
      "major": 9,
      "minor": 0,
      "patch": 0,
      "build": 27243,
      "released": false
    },
    "network": {
      "vpn": {},
      "stats": []
    }
  },
  "local_info": [],
  "rs232": [],
  "remote_setup": [],
  "schedule": {
    "name": "Audition",
    "type": "audition",
    "layout_href": "file:config/audition/audition.xml",
    "audio_muted": false,
    "screen_on": true
  }
}
```

Selected Player Status Information

URI

```
/JSON1/player/14533/status?indent=2&fields=system,schedule
```

Response


```
{
  "serial_number": 29926,
  "player_name": null,
  "reported_at": "2011-10-11T10:02:05+01:00",
  "system": {
    "up_since": "2011-10-06T15:23:53+01:00",
    "hardware": {
      "model": "NTB655",
      "oem_model": "NTB655P",
      "disk_temperature": 34.0,
      "disk_safety_margin": 21.0,
      "cpu_temperature": 50.0,
      "cpu_safety_margin": 45.0,
      "cpu_fan_speed": null,
      "system_fan_speed": 2481,
      "system_temperature": 41.0,
      "data_disk": {
        "free": 145683009536,
        "total": 146841247744
      },
      "system_disk": {
        "free": 2067959808,
        "total": 4227530752
      }
    },
    "software": {
      "major": 9,
      "minor": 0, 334
      "patch": 0,
      "build": 27243,
      "released": false
    },
    "network": {
      "vpn": {},
      "stats": []
    }
  },
  "schedule": {
    "name": "Audition",
    "type": "audition",
    "layout_href": "file:config/audition/audition.xml",
    "audio_muted": false,
    "screen_on": true
  }
}
```

3.3 Listing all Organisations

URI

```
/JSON1/organisations/all
```

Permission Required

API System Administration

Examples

URI

```
/JSON1/organisations/all?indent=2&callback=decodeAllOrganisations
```

Response

```
decodeAllOrganisations({
  "all_organisations": [
    {
      "name": "org1",
      "minimum_reporting_interval": null,
      "allow_alarm_ack": true,
      "allow_alarm_email": true,
      "allow_alarm_config": true,
      "allow_thumbnail_reporting": true,
      "allow_rs232_reporting": true,
      "player_limit": null,
      "reports_expiry_age": 90,
      "mae_expiry_age": 90
    },
    {
      "name": "org2",
      "minimum_reporting_interval": null,
      "allow_alarm_ack": true,
      "allow_alarm_email": true,
      "allow_alarm_config": true,
      "allow_thumbnail_reporting": true,
      "allow_rs232_reporting": true,
      "player_limit": null,
      "reports_expiry_age": 90,
      "mae_expiry_age": 90
    }
  ]
})
```

ONELAN Digital Signage

Part IV

Database Views

4 Database Views

The database views API provides direct access to the media audit event data and alarm information held in the DSM databases.

To examine the databases, you need to use a DSM account with the **External Database Reports** permission.

Use the database views to:

- Access media audit events 39.
- Access alarm information 39.

4.1 Accessing Media Audit Events

Use the `media_played_audit` view to access media audit event data.

Following is the list of fields within which the DSM holds the data:

Field	Description
`event_id`	A unique identifier for the media played event.
`player_serial`	Serial number of the player that this event belongs to.
`channel_id`	A unique identifier for the channel content.
`channel_name`	Name of the channel that the media came from.
`media_name`	The media item name.
`start_time`	Time the media started playing.
`duration`	Length of time the media played.
`end_time`	Time the media stopped playing.

4.2 Accessing Alarm Information

Use the `alarm_info` view to access alarm information for all of the user's Organisations.

Following is the list of fields within which the DSM holds the data:

Field	Description
`player_serial`	Serial number of the player for which DSM raises the alarm.
`alarm_name`	Name of the alarm.
`alarm_instance`	The unique alarm identifier.
`alarm_group_name`	Name of the alarm group the alarm belongs to

ONELAN Digital Signage

Field	Description
raise_time	Time DSM raised the alarm.
ack_time	Time the alarm was acknowledged.
lower_time	Time DSM lowered the alarm..
organisation_name	Name of the Organisation the player belongs to.