ONVIF™
Application Management Service Specification

Version 22.12

December, 2022

CONTENTS

1 Scope

This document defines an interface for managing applications on a device. This specification focuses on the network interface for managing the application lifecycle on a device. It includes installation, information retrieval as well as controlling execution. The actual runtime environment as well as the application packaging format are out of scope of this specification.

2 Normative references

ONVIF Core Specification
<http://www.onvif.org/onvif/specs/core/ONVIF-Core-Specification.pdf>

3 Informative references

Docker Enterprise Application Container Platform
< https://www.docker.com/ >

Android
< https://www.android.com/ >

4 Terms and Definitions

4.1 Definitions

App	Application that can be installed and execute on a networked device.

4.2 Abbreviations

HTTP	Hypertext Transfer Protocol
TLS	Transport Layer Security

5 Overview

A variety of standardized and proprietary application runtime formats exist. Examples for well-known standardized runtimes are Docker and Android. This specification provides network interface APIs based on the ONVIF Core Specification for uploading and controlling applications on a networked device.

The following basic functionalities are defined by this specification:

- Installing and uninstalling of applications.
- Starting and stopping execution of applications.
- Enumeration of installed applications
- Listing of application details
- Enrolling application licenses

An application state change event allows client to retrieve and track application state changes without need for polling.

Note that both the container format as well as the runtime are outside of the scope of this specification.

6 Service

6.1 General

Each application has an associated state. Possible states are Installing, Active, Inactive, DeInstalling and InstallationFailed. Clients can retrieve the application state via the GetAppInfo method or by subscribing to the application state property event. The latter mechanism allows retrieving the initial state as well as updates via property events.

6.2 Application Deployment

6.2.1 Install

Applications are deployed to a device via http POST to the URI provided by GetServiceCapabilities. The same URI shall be used for updating an application.

A device supporting this service shall support the POST operation to the upload URI provided by the UploadUri capability.

When posting an image to one of the URIs, the following HTML status codes may be returned, depending on success or failure:

- 200 OK

  Image file was successfully uploaded.

- 401 Unauthorized

  Attempted POST without authentication credentials at the WRITE_SYSTEM access policy level

- 408 Request Timeout

  POST took too long to upload

- 411 Length Required

  POST does not include Content-Length header

- 415 Unsupported Media Type

  POST does not include Content-Type header, or file MIME type does not match any type in the FormatsSupported attribute's list

Note that the installation may not be completed when the upload has finished.

If uploading succeeded, a device shall signal start of the installation via the state property event. If the installation fails, the device shall send a final InstallationFailed state. Otherwise the state shall change to either Active or Inactive.

6.2.2 Update

For updates, a device shall support the same mechanism as the installation procedure defined in 6.2.1.

In case an application is in state Active, the device shall change its state to Inactive before signaling installation state change events as stated in 6.2.1.

6.2.3 Uninstall

A device supporting this service shall support this method to stop and remove an installed application.

REQUEST:

- **AppID – [xs:string]**
  Application to be removed

RESPONSE:

<empty>

FAULTS:

- **env:Sender - ter:InvalidArgVal – ter:UnknownId**
  The requested application ID does not exist.

ACCESS CLASS:

SYSTEM_WRITE

6.2.4 Activate

A device supporting this service shall support this method to start an installed application.

REQUEST:

- **AppID – [xs:string]**
  Application to be started

RESPONSE:

<empty>

FAULTS:

- **env:Sender - ter:InvalidArgVal – ter:UnknownId**
  The requested application ID does not exist.

ACCESS CLASS:

SYSTEM_WRITE

6.2.5 Deactivate

A device supporting this service shall support this method to stop an installed application.

REQUEST:

- **AppID – [xs:string]**
  Application to be stopped

RESPONSE:

<empty>

FAULTS:

- **env:Sender - ter:InvalidArgVal – ter:UnknownId**
  The requested application ID does not exist.

ACCESS CLASS:

SYSTEM_WRITE

6.2.6 Backup and Restore

A device signals support for backup and restore of individual applications by providing the Configuration URI in the application information items (see 6.3). Clients can retrieve and backup the current configuration via

the http GET method. Note that the configuration itself is an opaque blob. A backed-up configuration may be restored at a later time by uploading it to the device via the http POST method. Refer to the individual application documentation regarding support for restoring of configurations across devices and versions. For general error behavior of the restore function see 6.2.1. Additionally an application may generate the following error response

- 521 Version mismatch The application cannot restore the provided configuration.

6.2.7 Application Interface

A device should provide an interface description for applications providing a programming API for configuration and monitoring purposes. For these applications a device should provide a link via the InterfaceDescription information item as defined in 6.3. It allows clients to easily integrate application configuration and information retrieval. Examples for interface description languages are JSON or YAML descriptions according to OpenAPI.

6.3 Application Information

6.3.1 Information Items

The following items may be associated to each application and can be queried either via the GetInstalledApps method or the GetAppInfo method.

AppID	Unique identifier of an application
Name	User readable name of the application
Version	Version string assigned by the application vendor
Licenses	Licenses associated with the application
Privileges	Privileges assigned to the application
InstallationDate	Date when the application was installed
LastUpdate	Date when the application was last updated
State	Information whether the application is currently Installing, Active, Inactive or Uninstalling (see Figure 1). InstallationFailed state can be provided only in AppMgmt/State event.
Status	Textual description of the state. In error cases this shall contain the error reason.
Autostart	Indication whether the application automatically starts on system boot.
Website	Link to an internal or external website providing supplementary information about the application or its vendor.
Configuration	Optional Uri for backup and restore of the application configuration.
InterfaceDescription	Optional reference to the applications interface definition.
OpenSource	Optional link to a listing of related open source licenses.

6.3.2 GetInstalledApps

A device supporting this service shall support this command to enumerate all installed applications. This method provides only high level information on the applications for more detailed information see GetAppInfo.

REQUEST:

<empty>

RESPONSE:

- **App – optional, unbounded [Application]**
  List of applications containing its ID as well as a user readable name.

FAULTS:

<none>

ACCESS CLASS:

SYSTEM_READ

6.3.3 GetAppsInfo

A device supporting this service shall support this command to retrieve detailed information about applications. The caller may provide an application ID to retrieve the information for a single application. If no application ID is provided, the device shall report the information for all installed applications.

REQUEST:

- **AppID – optional [xs:string]**
  Optional ID to only retrieve information for a single application

RESPONSE:

- **Info – optional, unbounded [AppInfo]**
  List of detailed information about the applications.

FAULTS:

- **env:Sender - ter:InvalidArgVal – ter:UnknownId**
  The requested application ID does not exist.

ACCESS CLASS:

SYSTEM_READ

6.4 Licensing

6.4.1 InstallLicense

A device signaling support via the Licensing capability shall support this command to install a license on the device. The format and structure of the license string is outside of the scope of this specification.

Refer to GetAppInfo for listing installed application licenses.

REQUEST:

- **AppID – optional [xs:string]**
  Application the license shall be associated to.
- **License – [xs:string]**
  Computer readable license string to be installed.

RESPONSE:

<empty>

FAULTS:

- **env:Sender - ter:InvalidArgVal – ter:InvalidFormat**
  The device cannot interpret the license either because the format is unknown or because it has invalid syntax.

ACCESS CLASS:

SYSTEM_WRITE

6.5 GetServiceCapabilities

The capabilities reflect optional functions and functionality of a service. The information is static and does not change during device operation.

REQUEST:

This message is empty

RESPONSE:

- **Capabilities [Capabilities]**
  Set of indicators for function groups as described above.

FAULTS:

None

ACCESS CLASS:

PRE_AUTH

The following capabilities are available:

UploadUri Mandatory upload URI for applications.

Licensing Signals support for licensing of applications.

SupportedFormat List of application formats supported by the device.

EventTopicPrefix Optional Event Topic prefix used when delivering app events in eventservice.

7 Events

7.1 State Change

Whenever the state of an application instance changes the following property event shall be generated. Figure 1 shows the possible states and their transitions.

tns1:AppMgmt/State

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="AppID" Type="xs:string"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="ans:AppState"/>
    <tt:SimpleItemDescription Name="Status" Type="xs:string"/>
  </tt:Data>
</tt:MessageDescription>
```

The Status item is optional. It shall be included whenever an error triggered the state change.

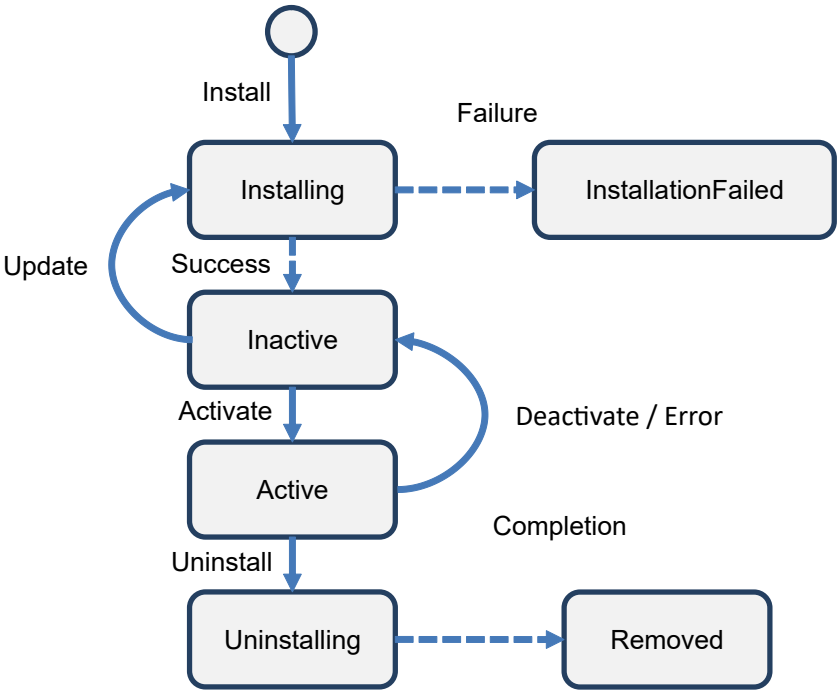

Figure 1: Application state chart.

7.2 Events From App

Based on app's internal configuration (outside the scope of onvif), app generated event can be either delivered with a standard event topic or can be delivered with prefix like "tns1:{EvenTopicPrefix}/{AppID}/{EventName}".

For example, if EventTopicPrefix is notified as "AppEvent", AppID is "DetectorApp" and event name is "FaceDetection", then the topic would look like tns1:AppEvent/DetectorApp/FaceDetection.

Client interested in events from installed app, can get the updated list of supported events using event service "geteventproperties" api response, whenever there is change in "tns1:AppMgmt/State" event.

Annex A.
Revision History

Rev.	Date	Editor	Changes
19.12	Dec-2019	Hans Busch	Initial release
21.06	Jun-2021	Hans Busch	Update backup and restore. Add configuration URI.
21.12	Dec-2021	Fredrik Svensson	Fix inconsistencies between appmgmt schema and doc.
22.06	Jun-2022	Sujith Raman	Correct namespace of event State member.
22.12	Dec-2022	Sujith Raman	Update Appmgmt app state enumeration.