

---

# APS Licensing Aspect

1.2

Copyright © 1999, 2009 Parallels, Inc

All rights reserved.

## Table of Contents

1. Introduction .....	1
2. Declarations .....	1
2.1. Metadata declaration .....	1
2.2. Schema of activation data .....	2
3. License manipulation script .....	3
4. Environment variables .....	3
5. Processing .....	3

## 1. Introduction

This APS package format aspect describes APS interface for licensing applications.

Licensing APS applications is performed by means of license keys. Each application instance or service may require license key(s).

## 2. Declarations

### 2.1. Metadata declaration

RELAX NG schema of licensing aspect:

```
default namespace l = "http://apstandard.com/ns/1/licensing"
namespace      sa = "http://apstandard.com/ns/1"

## Requirement definition for licensing aspect
Requirement |= element license {

    ## ID of the license, used to identify license in scripts
    attribute id { xsd:anyURI }?,

    ## License type URI
    attribute type { text },

    ## To issue valid license ISV requires to query application first
    element need-query { empty }?,

    ## Application will work without license
    element optional { empty }?

} *

## Provision method definition
ProvisionMethod |= element license-script {
    ## script name under scripts/ directory
    attribute name { text },

    ## if true script should be executed under privileged user
    attribute privileged { "true" }?,

    ## Script language or direct reference to binary executable
    ( element sa:script-language { text } |
      element sa:binary-executable { empty } )
}
```

```
}

```

```
<l:license id="main" type="http://my.company.com/application/license/uri">
  <l:need-query/>
  <l:optional/>
</l:license>

```

If web application needs the license one should include element declared above into the metadata in the `sa:requirements` section. Attribute `type` denotes unique license type identifier. Controller requests or matches available licenses using this type with license authority. `optional` element allows running application without valid license, for example in a demo mode. `need-query` element specifies that application should be queried before license request is sent.

Element `license` must be declared only as a top-level element of the `sa:requirements` section.

```
<l:license-script name="set-license.php">
  <script-language>php</script-language>
</l:license-script>

```

To define procedure of license installation, a license manipulation script must be declared in the `sa:provision` section. It works exactly as `sa:configuration-script`.

Element `l:license-script` must be declared only as a top-level element of the `sa:provision` section.

## 2.2. Schema of activation data

RELAX NG schema of activation data file:

```
default namespace a = "http://apstandard.com/ns/1/licensing/activation-data"
namespace l = "http://apstandard.com/ns/1/licensing"
namespace sa = "http://apstandard.com/ns/1"
namespace local = ""

## This part is related to aspect schema of license activation data file

grammar {

  start = ActivationData

  definedByApplication = element * - ( sa:* | l:* | a:* | local:* ) {
    attribute * { text }*,
    ( text | definedByApplication )*
  }

  ActivationData = element activation {

    ## URI element of application site to bound license to
    element uri { xsd:anyURI } ?,

    ## IP address of site where application is deployed
    element ip-address { text } *,

    ## Hostname of site where application is deployed
    element hostname { text } *,

    ## Domain name where application hosted
    element domain { text } *,

    definedByApplication *
  }
}
```

Activation data file returned by query operation should match above schema. Application packager may add any other relevant fields under `definedByApplication` part.

Activation data example:

```
<a:activation xmlns:a="http://apstandard.com/ns/1/licensing/activation-data">
  <a:uri>http://my.site.com</a:uri>
  <a:ip-address>10.20.30.40</a:ip-address>
  <a:hostname>srv1.hosting.com</a:hostname>
  <a:domain>site.com</a:domain>
  <unittest:testhash xmlns:unittest="http://www.unittest.com">34567890-09876534567</unittest:testhash>
</a:activation>
```

### 3. License manipulation script

License manipulation script declared with `license-script` element in provision section should meet requirements defined for the `configuration-script` element in the Specification. It should get instance-specific license information and provide an ability to install and remove license.

This script is to be invoked every time when new license is issued for the application, old license is updated or removed.

Script should be invoked with the following arguments and perform respective actions:

- `install` - install new license, license file to be provided in `LICENSE_<id>_FILE` file. Where `<id>` refers to license identifier optionally provided on `license` element.
- `remove` - remove installed license. This action is to be performed in case of license cancellation by issuer.
- `query` - get information required for license issuing. Information is to be stored in `LICENSE_<id>_DATA` file. Where `<id>` refers to license identifier optionally provided on `license` element. This file may include URI of site where application is to be executed, environment information, hardware binding, etc.

### 4. Environment variables

All information about application, resources and settings is passed to license script through environment variables.

Same rules and set of variables are used for license script as for configuration script.

Environment variable `LICENSE_LIST` must be passed to the `license-script` script on any operation. It should contain space-delemited list of licenses IDs involved in the operation.

Environment variable `LICENSE_<id>_FILE` must be passed to the `license-script` script when license is first installed or updated. This variable must contain full path to the file with the license key. The script must be executed under account with permission to read this file. This variable (with respective ID) should be provided for every license key involved in operation.

Environment variable `LICENSE_<id>_DATA` must be passed to the `license-script` script when instance is queried for license activation data. Application should write environment-specific data to that file, and then file will be passed by Controller to license issuer for processing. The script must be executed under account with permission to write in this file. Data written to file should match `activation.rnc` schema. This variable (with respective ID) should be provided for every license key involved in operation.

### 5. Processing

Controller SHOULD process application query result to form request to license authority (i.e. KA). When license is issued Controller should install license to application or upgrade installed one.

Controller must provide license(s) for applicaton according to policy of the `optional` element:

- `optional` - If the `l:optional` element is present an application may be provided to end-user without valid license.
- `mandatory` - If the `l:optional` element is absent an application valid license should be installed before application becomes available for end-user. This means that after application instance creation and before license is installed application instance should exist but control panel should not allow to access to application.

Application is supposed to check the validity of the license (KA signature, validity date, installation URL, other parameters) and behave accordingly.

It is not guaranteed that license will be provided to the application immediately after the installation, so applications should expect some timeframe of operation without license, by, e.g. denying access to the itself or by providing demo mode. Control panel should provide appropriate hint in case license is not optional.