App Description Specification

Version 001

PLT-Bericht

Dresden, 01.04.2015

Document history

Version	Date	Author	Comment
001	2015-04-01	Johannes Pfeffer	Initial version of the specification

Abstract

This document specifies the structure and content of App-Descriptions for App-Orchestration. The goal is to have a reference specification that all creators of App-Descriptions can refer to.

Contacts

Dipl.-Ing. Johannes Pfeffer is responsible for this document.

E-Mail: johannes.pfeffer@tu-dresden.de

Phone: +49 351 463 33387

At a glance

An App-Description semantically describes an app. It at least contains the following information: about the name of the app, a textual description, the creator, the type of the app, location of binaries and screenshots, entry points and exit points.

App-Descriptions are modeled in W3C's RDF-format, usually in turtle notation.

App-Descriptions adhere to the open-world assumption, meaning there is no limit to what can be asserted about an app. However, this specification defines a minimum of information that must be given and provides a recommendation on additional information to provide about an app.

App-Descriptions are used in App-Ensembles (see App-Ensemble specification) to provide information about an app, such as available inputs and outputs.

Specification

Introduction

App-Descriptions are described as RDF and are usually written (serialized) in Turtle-format. They are named according to the following schema:

```
NameOfApp.ttl
```

An App-Description model consists of the following main parts:

- Definition of prefixes
- Identifier
- Information about the role of the app
- Type and role of the app
- Label and textual description
- Binary location and version
- Dependencies
- Information about the creator(s) of the app
- Entry-Points (with inputs)
- Exit-Points (with outputs)
- Icon and screenshots
- Additional statements about the app

Used Vocabularies & ontologies

The main vocabulary for App-Descriptions is the *Application-Orchestration-Framework-Vocabulary* with the preferred prefix `aof` and the base URI http://eatld.et.tu-dresden.de/aof/.

```
@prefix aof: <http://eatld.et.tu-dresden.de/aof/> .
```

For describing the creator and her/his contact information, the *Dublin-Core Metadata Format* and the *FOAF RDF-Schema* are used.

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

To describe android-related properties of an app the Android apk namespace is used.

```
@prefix android: <http://schemas.android.com/apk/res/android> .
```

In addition the rdf, rdfs schemas are used as a basic language and the xsd schema is used to define value types.

Definition of Prefixes

It is good practice to add a section with prefixes used in the App-Description.

This section may look like the following listing:

```
# AOF namespace
@prefix aof: <http://eatld.et.tu-dresden.de/aof/> .

# Used ontologies
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix android: <http://schemas.android.com/apk/res/android> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

In addition, it has proven useful to define some namespaces that may be used only in a certain App-Ensemble. For example if the app is built and located on a Jenkins build-server, you may add:

```
# Namespaces for THIS App-Ensemble
@prefix jenkins: <http://dev.plt.et.tu-dresden.de:8085/jenkins/job/> .
@prefix ex: <http://example/
```

Identifier

An App-Description always starts with the following triple:

```
<http://url_that_uniquely_identifies_the_app> a aof:App .
```

The subject of this statement is important because it is the main identifier that is used by tools that work with App-Descriptions. It should be long-living and if possible resolve to information about the app. Possibilities are the URL to an App-Store where the app can be downloaded or to a job on a build-server that creates the app.

In this specification we will use ex-prefix (`<http://example/>`) and the app name *ExApp* for all examples.

Type and role of the App

There are two bits of information that should be provided about an app. First the `rdf:type` relation[1] and second the `aof:hasRole` relation. Both are *optional*.

All apps are of type `aof:App`. Additionally, Android apps are of type `aof:AndroidApp`. Other platforms are not yet included in this specification. Please note, while this statement is optional it is highly recommended because applications may use it to determine the platform of the app.

```
ex:ExApp a aof:App, aof:AndroidApp .
```

There are some special roles for apps that are important for App-Ensembles. Normal apps don't need this statement.

```
ex:ExApp aof:hasRole aof:Conductor .
```

The `aof:Conductor` role means that the app can execute App-Ensembles by itself. Also available is `aof:AppEnsembleInstaller` which implies that the app is able to install App-Ensembles on a platform.

Label and textual description

All apps must have a label (`rdfs:label`) and description (`rdfs:comment`).

```
ex:ExApp rdfs:label "App-Description Specification Example App" ;
  rdfs:comment """
    The rdfs:comment should give a short textual description
    about the functionality of the app. It may contain *markdown* markup """ .
```

[1]The `rdf:type` relation can be abbreviated by `a` in Turtle notation

Binary location and version

If available, the location of a resource where the current binary can be downloaded should be given via the `aof:currentBinary` property. Preferably, this is a direct link to a binary. This property is optional but highly recommended.

```
ex:ExApp aof:currentBinary <http://example/example.apk> .
```

The version number must be an integer. Version information may be provided in two different ways. The first option is to provide an http-URL to a resource that delivers the version code in the body part of the http response using `aof:versionCodeReference`. The second way is to provide a version code integer directly via `aof:versionCode`. The `aof:versionCodeReference` variant supersedes `aof:versionCode`.

```
ex:ExApp aof:versionCodeReference jenkins:AppEnsembleInstaller/buildNumber .

# The alternative:
ex:ExApp aof:versionCode "1"^^xsd:integer .
```

Dependencies

If the app has dependencies in other apps, i.e. these other apps are required for it to function properly, they may be stated using `aof:hasAppDependency`.

```
ex:ExApp aof:hasAppDependency <http://openintents.org/~oi/filemanager> .
```

The object of this statement should be a URI that is unique and is used in other App-Descriptions when referring to this app.

Creator

Information about the creator(s) is added by asserting one or more of the following statements:

```
ex:ExApp dc:creator <mailto:max.mustermann@example.org> .
```

The object of a `dc:creator` relation is a URI that uniquely identifies a creator (e.g. an e-mail address or a website). At the end of the App-Description resource should be detailed using the FOAF-vocabulary:

```
<mailto:max.mustermann@example.org> a foaf:Person ;
  foaf:name "Max Mustermann" ;
  foaf:mbox <mailto:max.mustermann@example.org> ;
  foaf:homepage <http://maxmustermann.info/> .
```

Creator information is optional but highly recommended.

Entry Points

Entry points provide information on how the app is started and what information it can use as input. There must be at least one entry point. Currently, entry points are only specified for Android apps.

A basic entry point looks like this:

```
ex:ExApp aof:providesEntryPoint [
  a aof:EntryPoint, android:action ;
  android:name "org.aof.action.EXAMPLE_ACTION" ;
  rdfs:label "Example label" ;
  rdfs:comment "
    The comment should give a concise but conclusive description about what the
    entry point does. """ ;
] .
```

An entry point may have zero or more inputs. If an entry point is given `aof:Input, android:extra, rdfs:comment, aof:isRequired, android:name` and `aof:datatype` are all mandatory.

```
ex:entry_point_1 aof:hasInput [
  a aof:Input, android:extra ;
  rdfs:comment "Description of the input" ;
  aof:isRequired "False" ;
  android:name "org.aof.extra.POI" ;
  aof:datatype xsd:anyURI
] .
```

The statement `ex:entry_point_1 a aof:Input, android:extra` asserts that the resource is an entry point and also an android extra. Currently, only this type of input is specified and, therefore, must be provided if entry points are given. `aof:isRequired` states if this input must be provided, `android:name` gives the name of the extra, i.e. the key to look up the extra data field. There may be a naming convention for `android:name` that should be followed. For details see the App-Ensemble specification. The property `aof:datatype` states the data type the extra must have using the xsd: notation.

Exit points

Exit points provide information on data that is generated by an app. When apps provide information on exit points it is assumed that they have been created specifically for App-Orchestration and provide the output data after an orchestration specific user interaction, such as a *Done-Button.*

Currently exit points are only specified for Android apps.

An exit point must have at least one output. All properties in the example below are mandatory if an exit point is described.

A typical exit point looks like this:

```
ex:ExApp aof:providesExitPoint [
  a aof:ExitPoint;
  rdfs:label "Main exit point" ;
  rdfs:comment "Description of the exit point" ;
  aof:hasOutput [
    a aof:Output, android:extra ;
    rdfs:comment "Description of the output" ;
    aof:isGuaranteed "False" ;
    android:name "org.aof.extra.POI" ;
    aof:datatype xsd:anyURI
  ]
] .
```

The property `aof:isGuaranteed` is used to specify if an output will always be available or not. The property `android:name` is used to give the key for the data field. For the other properties, please see the section on entry points. There may be a naming convention for `android:name` that should be followed. For details see the App-Ensemble specification.

Icon and screenshots

It is recommended to provide a URI of an icon and some screenshots for all apps because this improves the display in lists, app-stores, etc. greatly. The icon is optional (`aof:hasIcon`). There may be 0 or 1 *main* screenshot (`aof:MainScreenshot`) and 0 or more additional screenshots (`aof:Screenshot`).

Example:

```
# Icon
ex:exApp aof:hasIcon <http://example/icon_placeholder.svg> .

# Screenshots: 1 main screenshot, 2 other screenshots
```

```
ex:exApp aof:MainScreenshot [
  aof:hasScreenshot <http://example/sc1.jpg> ;
  aof:hasScreenshotThumbnail <http://example/sc1_thmb.jpg> ;
  rdfs:comment "Main screenshot"
] .
ex:exApp aof:Screenshot [
  aof:hasScreenshot <http://example/sc2.jpg> ;
  aof:hasScreenshotThumbnail <http://example/sc2_thmb.jpg> ;
  rdfs:comment "2nd screenshot"
] .
ex:exApp aof:Screenshot [
  aof:hasScreenshot <http://example/sc3.jpg> ;
  aof:hasScreenshotThumbnail <http://sc3_thmb.jpg> ;
  rdfs:comment "3rd screenshot"
] .
```

Additional statements

Since an App-Description lives in the open world assumption, there may be unlimited other statements about an app. Here are some examples:

```
# References to resources describing the same thing or related things
ex:ExApp owl:sameAs <http://otherExample.org/app > .
ex:ExApp rdfs:seeAlso <http://alternativeAppStore/otherApp .
```

Examples

The following examples provide complete App-Descriptions that may be used as a starting point.

Conductor App

This example describes a conductor app, i.e. an app that executes App-Ensembles. Therefore, it has a `aof:hasRole aof:Conductor` statement.

```
# AOF prefixes
@prefix aof: <http://eatld.et.tu-dresden.de/aof/> .

# Used Ontologies
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix android: <http://schemas.android.com/apk/res/android> .
```

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# Namespaces just for THIS App-Description
@prefix jenkins: <http://dev.plt.et.tu-dresden.de:8085/jenkins/job/> .

# App description
jenkins:ExampleConductor
  a aof:App, aof:AndroidApp ;
  aof:hasRole aof:Conductor ;

  # Name and textual description
  rdfs:label "AOF Conductor" ;
  rdfs:comment """
    The *Conductor* is an app that executes App-Ensembles on Android devices.
    """ ;

  # Reference to binary and version information
  aof:currentBinary <http://example/Conductor.apk> ;

  # This App-Description provides the versionCode as a http-resource
  aof:versionCodeReference <http://Example/buildNumber> ;
  # The alternative:
  # aof:versionCode "1"^^xsd:integer ;

  # Dependencies
  aof:hasAppDependency play:org.openintents.filemanager ;

  # Creator reference, this is detailed at the end of the App-Description
  dc:creator <mailto:johannes.pfeffer@tu-dresden.de> ;

  # Entry points
  aof:providesEntryPoint [
    a aof:EntryPoint, android:action ;
    android:name "org.aof.action.EXAMPLE_ACTION" ;
    rdfs:label "Start Workflow" ;
    rdfs:comment """
      The comment should give a concise but conclusive description about what the
      entry point does. """ ;
    aof:hasInput [
      a aof:Input, android:extra ;
      rdfs:comment "Description of the input" ;
      aof:isRequired "False" ;
      android:name "org.aof.extra.POI" ;
      aof:datatype xsd:anyURI
    ]
  ] ;

  # Exit points
  aof:providesExitPoint [
    a aof:ExitPoint;
    rdfs:label "Main exit point" ;
    rdfs:comment "Description of the exit point" ;
    aof:hasOutput [
      a aof:Output, android:extra ;
      rdfs:comment "Description of the output" ;
      aof:isGuaranteed "False" ;
      android:name "org.aof.extra.POI" ;
      aof:datatype xsd:anyURI
    ]
  ] ;

  # Icon
  aof:hasIcon <http://example/icon_placeholder.svg> ;

  # Screenshots: 1 main screenshot, 2 other screenshots
  aof:MainScreenshot [
    aof:hasScreenshot <http://example/sc1.jpg> ;
    aof:hasScreenshotThumbnail <http://example/sc1_thmb.jpg> ;
    rdfs:comment "Main screenshot"
    ] ;
  aof:Screenshot [
    aof:hasScreenshot <http://example/sc2.jpg> ;
    aof:hasScreenshotThumbnail <http://example/sc2_thmb.jpg> ;
    rdfs:comment "2nd screenshot"
    ] ;
  aof:Screenshot [
    aof:hasScreenshot <http://example/sc3.jpg> ;
    aof:hasScreenshotThumbnail <http://sc3_thmb.jpg> ;
    rdfs:comment "3rd screenshot"
    ] ;

# Additional statements
# Creator
<mailto:johannes.pfeffer@tu-dresden.de> a foaf:Person ;
  foaf:name "Johannes Pfeffer";
  foaf:mbox <mailto:johannes.pfeffer@tu-dresden.de>;
  foaf:homepage <http://www.et.tu-dresden.de/ifa/index.php?id=jp> .

# References to resources describing the same thing or related things
```

```
jenkins:ExampleConductor owl:sameAs <http://appStore/Conductor> .
jenkins:ExampleConductor rdfs:seeAlso <http://alternativeAppStore/OtherConductor .
```

Workflows

Typical lifecycle of App-descriptions

Already during specification of an app a rudimentary app description with entry and exit points can be created. The URI of the app should be fixed already. Then, the App-Description should be updated with every public release of the app.

integrating app-descriptions into the build-process

It may be sensible to integrate publication of the App-Description in the build process of the app. E.g. the App-Description may reside in the project structure for the app and is automatically published to an web server during a release or build process.

Glossary

For an always up-to-date version of the glossary refer to:

https://eatplt02.et.tu-dresden.de/intrawiki/doku.php?id=wiki:documentation:aof:glossary