



# Policy



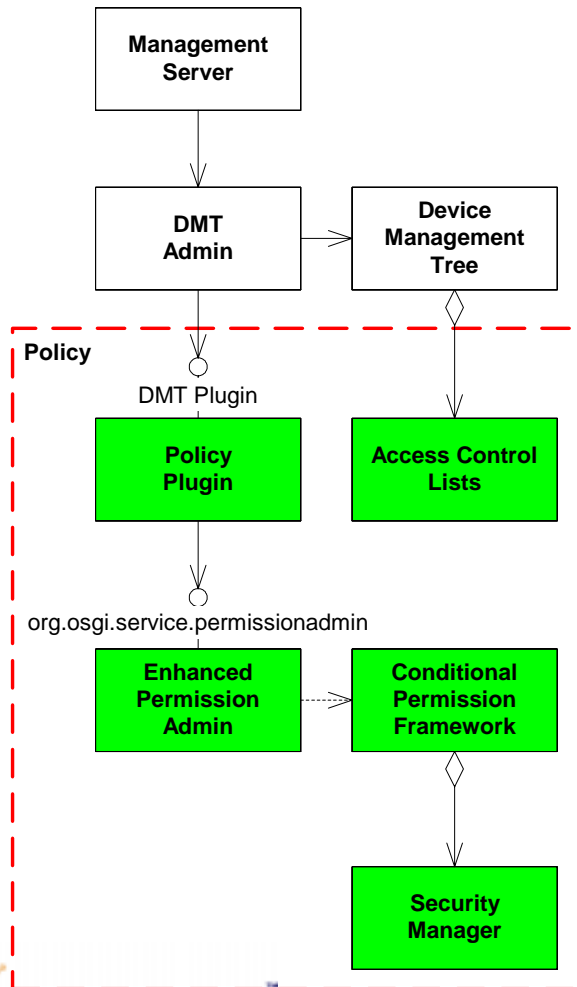
# Requirements for Policy

- Use cases and requirements:
  - Policy controlling the management system
    - Accept/reject various operations based on e.g. actor, target and other conditions
  - Controlling the runtime behaviors of applications
    - Allow/deny sensitive operations (assign privileges) for the applications
  - Delegation model
    - Possibility to delegate (parts of) the policy forward
  - Policy tied to subscription
    - Changing e.g. SIM card or user identity changes the utilized policy
  - Programmatic Interface
    - Privilege checking, permission querying/setting, dynamic updating

# General overview

- MEG defines a management framework and an application model for OSGi based devices
- Therefore, policy has two aspects:
  - Access control for management operations
  - Runtime control of applications
- These two are solved using different mechanisms:
  - Policy for management operations is expressed in the form of Access Control Lists for coarse grained and Java permissions for fine grained access control
  - Applications are running under the standard Java access control – according to JDK 1.2 specification – extended with some OSGi and MEG specific permissions
- Policy is manageable via the MEG-defined device management framework
  - Policy can be updated runtime
- Conditional Permission Framework is used to provide additional flexibility to Java permissions

# Policy Architecture



- Device Management Tree is used as management object model
- Each node in DMT can have an associated Access Control List
- ACL is enforced by DMT Admin when performing remote management operation
  - DMT Admin may also use Java permissions for fine-grained access control
- Java permissions are also mapped to the DMT to enable management via the Policy Plug-in
- Permission Admin is modified to support conditional permissions
- Conditional Permissions require a special Security Manager to evaluate conditions

# DMT Access Control Lists

- Device Management Tree is the management object model defined by OMA Device Management specifications.
  - For details see: <http://www.openmobilealliance.org/tech/affiliates/syncml/syncmlindex.html>
- Each node of DMT may have an associated Access Control List (ACL)
- ACLs contains the five management operations permitted by OMA (Add, Get, Delete, Replace and Exec) and associated with each of them a list of management server identifiers, which are permitted to perform the corresponding operation
  - For example (the exaple is copied from OMA DM specification):  
Add=www.sonera.fi-8765&Delete=www.sonera.fi-8765&Replace=www.sonera.fi-8765+321\_ibm.com&Get=\*
- DMT Admin authenticates the management servers and based on its identity it enforces the policy described in the ACLs.

# DMT Fine-grained access control

- In addition to ACLs, fine-grained access control may be applied to remote management operations
- It is possible to associate Java permissions with the management server identifiers
- After authenticating management server, DMT Admin loads the associated Java permissions and management operations are performed using those permissions only.
- Fine-grained permissions checks are made by the management object targeted by the management operation.
- Support of the permission-based fine grain policy check is optional.

# Changes to R3 Permissions

- The security mechanisms used in OSGi Release 3 will be extended
- AdminPermission is too broad, it will be refined by adding a target and operation category parameters to it.
  - The target of permission will be a bundle
  - Admin operations will be classified into categories to enable finer policy definition.
    - Such categories are: “metadata”, “class”, “lifecycle” etc.
- Permission Admin will be updated:
  - To support conditional permissions
  - Enable associating permissions to a signer
- Bundles may request permissions listing them in META-INF/permission.perm file. The actual set of permissions for the bundle will be the intersection of this list and that enabled by the policy.

# Application Access Control

- Access control for applications is implemented using the standard Java security mechanisms
- Permissions may be associated with bundles based on the signer and/or the location of the bundle containing the application
- Permission Admin is used to maintain the policy
- Services use SecurityManager to check permissions
  - Usage of SecurityManager instead of AccessController is important to enable the verification of conditions associated with the permissions

# Conditional Permissions

- Conditional Permission Framework enables the association of certain conditions with permissions
  - Any number of conditions can be associated with a permission
- Such conditions include the IMEI of the device, the IMSI of the SIM in the device, current geographic location of the device, consent of the device user etc.
  - Consent of the user means that the user is prompted for allowing the requested operation.
  - Similarly to MIDlet security model, he/she has the option to deny the request or grant it for one shot, session lifetime or blanket (I.e. forever)
- The owner of a conditional permission can hold the corresponding privilege only if all the associated conditions are met.
- Conditions are represented by classes implementing them
- A special SecurityManager implementation is responsible for the evaluation of conditions prior to granting access to the requested resource.

# Policy for other application models

- Application containers for other application models (e.g. MIDlets, Xlets etc.) can be executed in the OSGi framework
- These containers are responsible for implementing the security measures specified in the corresponding application model
- They are also responsible for mapping their policy structure into DMT to enable remote management.



# Download and Deployment



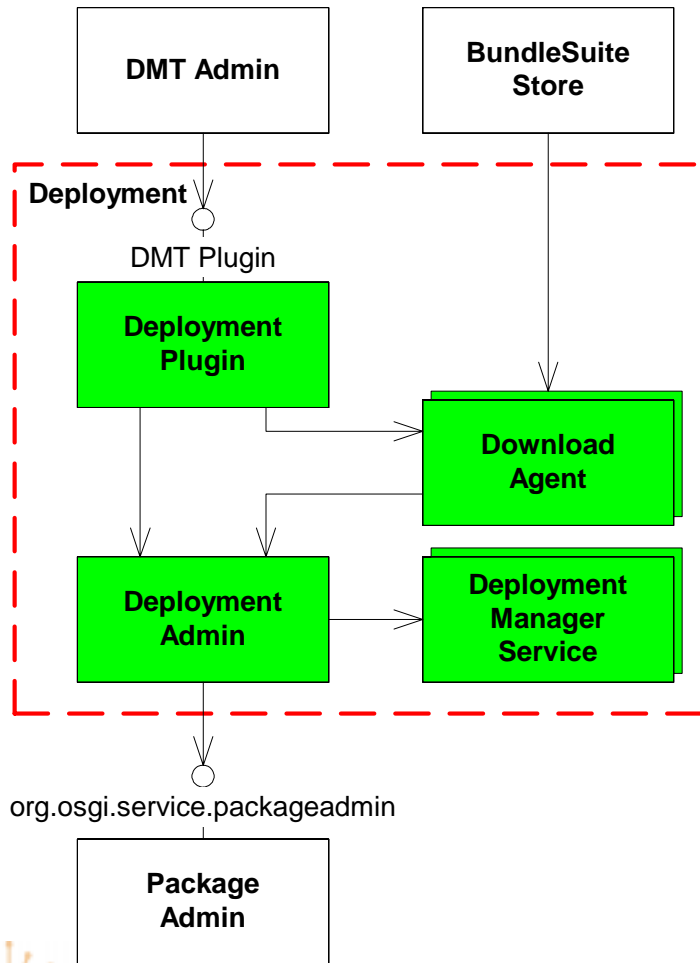
# Requirements for Deployment

- Use cases and requirements:
  - Installation/reinstallation/update/uninstallation
    - Multiple media, Dependency checks, related data, mass operations, demand loading
  - Integrity Checking
    - Checks the dependency tree for conflicts or missing pieces
  - Software Discovery
    - List of installed software components
  - Software subscriptions/unsubscriptions
    - Possibility to install software multiple times without paying it again
  - Controlled (un)availability of software
    - Possibility to make the software accessible on certain time different from deployment time
  - Operator characteristics
    - Software can be enabled/disabled or even installed/uninstalled based on the used operator (e.g. SIM dependently)

# General Overview

- MEG defines a packaging/install script format called bundle suite
- Bundle suites can contain OSGi bundles as well as other customization scripts, e.g. configuration information
- Installation of a bundle suite is atomic, it either succeeds completely or fails without side effect.
- Architecture supports different delivery media for bundle suites
- Bundle dependencies are tracked to ensure system integrity
  - In addition to the current OSGi dependencies (package and service dependency), bundle dependency is introduced to denote relations between bundles which are otherwise non-traceable.
- A bundle-level mechanism is defined, which enables the customization of bundles after installation
- Bundle upgrades are not used, new version is installed, old version is uninstalled instead.

# Deployment Architecture



- Deployment Plug-in maps deployment related management operations to managed object functions
- Download Agents are responsible for fetching bundle suites via different delivery media
- Deployment Admin is responsible for the installation of bundle suites
- Deployment Manager Services are used to process non-bundle elements of a bundle suite
- Deployment Admin relies on Package Admin when installing/removing a bundle

# Bundle Suites

- Bundle Suites are installation scripts used to install and customize a number of bundles together.
- Bundle Suites are JAR files. The order of the elements in the file is significant as it defines the processing order.
- The first entry in the file must be the META-INF/Manifest.mf
  - This file contains meta information to help the processing of the bundle suite
- Manifest.mf is followed by the signature files to enable the verification of the integrity of Manifest.mf and other files as well
- The remaining content may come in any order
  - JAR files are assumed to contain bundles
  - Other files as used for customization, e.g. configuration of bundles

# Deployment Admin

- Deployment Admin is responsible for the installation (execution) of bundle suites
- Deployment Admin processes the bundle suite element by element
  - JAR files are treated as bundles and are installed using Package Admin
  - Other files are passed to Deployment Manager Services for processing
- Deployment Admin verifies the integrity of the bundle suite by checking the digital signature of the package assembler on the suite
- Deployment Admin ensures that the installation of a bundle suite is atomic, i.e. if some step fails during the process, the entire transaction is rolled back without leaving any side effect.
- Deployment operation are accessible via Deployment DMT Plug-in.
- Deployment Admin implements a “garbage collection”-like solution to remove unnecessary bundles.
  - Deployment Admin tracks dependencies between bundles
  - Bundles may be marked as root. A non-root bundle is uninstalled if no root bundle depends on it.

# Deployment Manager Services

- Deployment Manager Services enable hooking in different customization operations to the processing of a bundle suite
- DMSs can be associated with filters on file names. During the installation of a bundle suite, when the name of an element matches some filters, the corresponding DMSs are invoked, passing the element to them.
- Any type of DMSs are possible; however, currently only two are standardized:
  - AUTOCONF.XML
    - This file contains configuration data. It should be processed by a DMS and using its content, configure bundles via Configuration Admin
    - The corresponding DMS is also responsible for proper book keeping of the created configuration items to ensure proper cleanup when the corresponding bundles are uninstalled.
  - Data customization in Manifest.mf:
    - The corresponding DMS extracts customization information from the bundle suite Manifest and executes customizers at the end of the installation.

# Download Agents

- Download Agents implement the support of different delivery media
  - They may work from a local memory card, connect to an installer on a PC or to a remote server
- Depending on the media type, Download Agents may be simple or may implement complex logic
  - For example they may send inventory of installed bundles of the device to enable the customization of bundle suites and saving bandwidth.
- A Download Work Stream was recently formed in MEG to define or select a standard download protocol

# Download Work Stream

- Goal: analyze existing technologies and their relevance to the OSGi/MEG specification in order to generate a guideline or even a specification to address the needs for download, security, distribution of DMO's and monitoring of OSGi/MEG enabled device over the air.
- The first step is to analyze technologies from OMA, as far as they are accessible to public, and JCP specifications JSR 118, 124 and 233. This will lead to a GAP analysis and helps to formulate the next steps.

# Thank you!

Gábor Pécsy

Software Technology Specialist

[gabor.pecsy@nokia.com](mailto:gabor.pecsy@nokia.com)