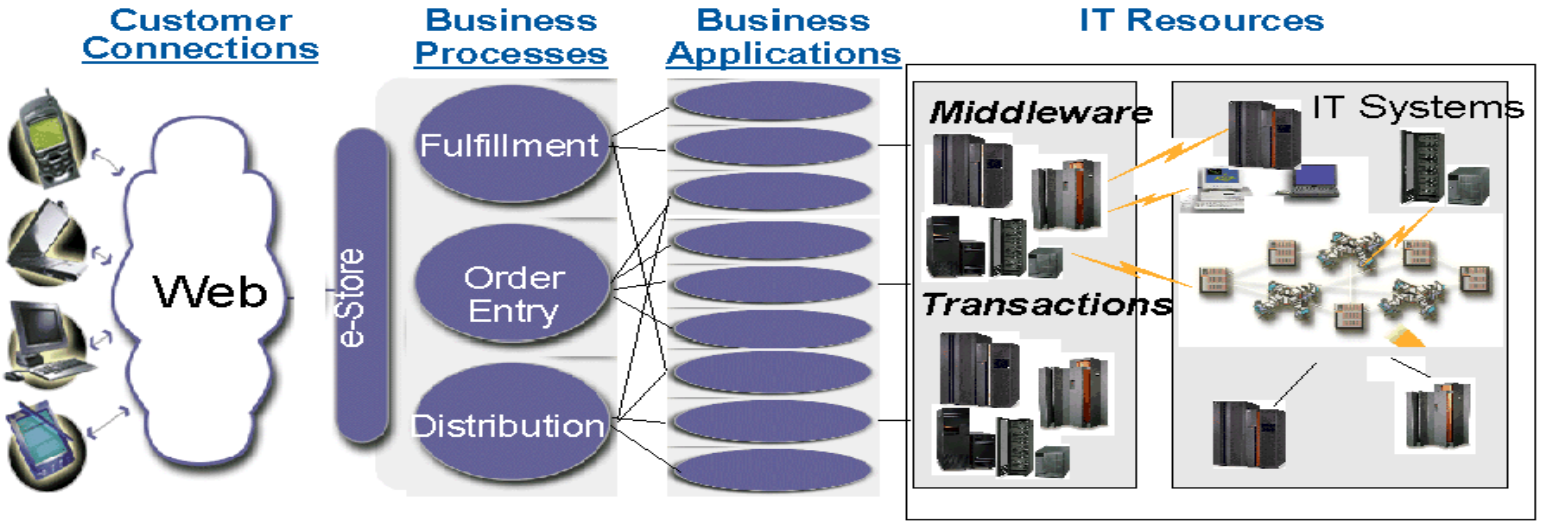
SDD Proposal to COSMOS

Jason Losh (SAS), Oasis SDD TC Tooling Lead
Mark Weitzel (IBM), COSMOS Architecture Team Lead

Why: *Customer Problems*

Customer Feedback – More than half of outages caused by inability to roll out application changes because of complex interdependencies with other application components and products



Increasing emphasis on "solutions"

- a combination of hardware and software components supporting a defined business process
- solutions need to be installed, configured, deployed, monitored, operated, problem diagnosed, maintained,...
- these tasks must be driven from a solution perspective
- the post-purchase experience shouldn't degenerate to a bunch of point products and solution components

Why: *Complexity of Managing Software*

Require external information to enable consumers to analyze and make pre-deployment decisions. Consumers can be tools that are performing composition or tools that are making pre-deployment decisions.

Package Identity

What is this thing?

What does it contain?

Requirements

What is needed to deploy this package?

What is needed to be maintained for the lifetime of the deployment?

Software Package
(Logical / Not Physical)

Package Variability

What "parts are needed to be deployed"?

Results

What does this provide?

What effect will this have on my environment?

Primary Use Cases

OASIS TC has defined use cases, including:

- Install to Development/Test
- Install to Production
- Aggregation/Complex Deployments
- Upgrade to new version
- Install patches/service packs
- Uninstall base, upgrade or maintenance

Source: http://www.oasis-open.org/committees/document.php?document_id=22893&wg_abbrev=sdd

These are the basic use cases required by products and customers as evidenced by OID and adopter requirements. They should be part of the first reference implementation.

PROPOSED COMMUNITY

- Many companies from the OASIS SDD TC and the industry have expressed interest in actively developing tools and runtime for the SDD standard
  - SAS
  - IBM
  - SAP
  - Wind River
  - Macrovision
- Most are expected to contribute at least one person to the project
- Others are expected to participate from COSMOS once it is socialized within the community

SDD Usage – Creation of SDDs

- SDDs that wrapper existing installs (such as ISMP, MSI, etc.) need to be created
  - Enhances adoption by leveraging existing inventory of install programs
- SDDs will need to be authored by solutions developers
  - Possible extension point – an IDE for editing/creating SDDs
- SDDs will need to be generated by build processes or post-build processes

SDD Usage – Consumption of SDDs

- SDDs will need to be processed and the instructions contained in them acted upon
  - Possible extension point – an SDD runtime must be extensible
- Metadata about changes that occur as a result of SDD processing needs to be stored for later retrieval for things like maintenance, hot fixes, etc.
  - Possible extension point – the underlying storage system should be capable of being overridden and/or extended
  - The implementation must not be bound to a specific resource model
- To support orchestrated installation of a distributed system, the runtime acting upon SDD needs to expose its availability to other consumers like an install program
- Install performance requirements:
  - Must be impacted at no more than 10% with minimal footprint and using small, nimble components (JRE, database, etc)
- Resource Models Requirements:
  - The runtime and tooling created in support of the SDD must be able to support different underlying resource models
- Security requirements:
  - Handle root/non-root installs, appropriate credentialing on remote install scenarios, and don't require writable space on install media/local disk

SDD Creation tools - Proposed development in open source

- **SDD Validator – checks syntax of the SDD XML**
  - Validator shall check CL1 based SDDs
  - Validator shall check CL2 based SDDs
- **SDD Programmatic Interface – deconstructs information in the SDD and creates objects from SDD elements**
  - Parser shall deconstruct CL1 based SDDs
  - Parser shall deconstruct CL2 based SDDs
  - SPI shall create data objects from SDD elements
  - SPI created data objects shall be consumable via API
- **SDD Wrapper Generator – creates SDD wrappers for existing install programs (MSI, ISMP, etc.)**
  - Wrapper Generator shall be capable of inspecting existing install packages and creating SDDs based upon them
  - Wrapper Generator shall generate CL1 based SDDs from existing install programs
  - Wrapper Generator shall generate CL2 based SDDs from existing install programs
- **SDD Build Time Generator – creates both CL1 singletons and CL2 composites**
  - BTG shall read SDD data objects and create CL1 SDDs
  - BTG shall read SDD data objects and create CL2 SDDs
  - BTG shall be “pluggable” into existing build environments

SDD Runtime - Proposed development in open source

- **SDD Runtime – reads instruction set encoded into SDD and acts upon those instructions such as resource constraints, environment checks, etc.**
  - Runtime shall support basic use cases described on slide five
  - Runtime shall be capable of acting upon the basic instruction set
  - Runtime shall perform at no less than 10-20% slower than native technology
  - Runtime shall take into account security requirements such as ACL, remote installation
- **SDD Runtime Embedded Storage – stores information about change to the software such as install, maintenance, hot fix, uninstall, etc.**
  - Embedded storage may be a relational database
  - Embedded storage shall house either a critical subset of SDD data or the SDD itself
  - Embedded storage shall provide a listing service for viewing of content
  - Embedded storage shall provide a service or utility for direct editing of content
  - Embedded storage shall support default and pluggable databases (use what already exists, e.g. CMDB)
- **SDD Service – serves as a local or remote reference to a runtime**
  - Service shall expose availability/capability of runtime(s)
  - Service shall keep an installation index up to date for easy lookup of deployed software
- **SDD Installation Index – an index for lookup of an SDD service**

Requirements for release: Target June 2008

- **SDD Validator – checks syntax of the SDD XML**
  - Validator shall check CL1 based SDDs
  - Validator shall check CL2 based SDDs
- **SDD Programmatic Interface – deconstructs information in the SDD and creates objects from SDD elements**
  - Parser shall deconstruct CL1 based SDDs
  - Parser shall deconstruct CL2 based SDDs
  - SPI shall create data objects from SDD elements
  - SPI created data objects shall be consumable via API
- **SDD Wrapper Generator – creates SDD wrappers for existing install programs (MSI, ISMP, etc.)**
  - Wrapper Generator shall be capable of inspecting existing install packages and creating SDDs based upon them
  - Wrapper Generator shall generate CL1 based SDDs from existing install programs
  - Wrapper Generator shall generate CL2 based SDDs from existing install programs
- **SDD Build Time Generator – creates both CL1 singletons and CL2 composites**
  - BTG shall read SDD data objects and create CL1 SDDs
  - BTG shall read SDD data objects and create CL2 SDDs
  - BTG shall be “pluggable” into existing build environments
  - BTG shall read SDD data objects and create CL1 SDDs
  - BTG shall read SDD data objects and create CL2 SDDs
  - BTG shall be “pluggable” into existing build environments

Requirements for release: Target June 2008

- **SDD Runtime – reads instruction set encoded into SDD and acts upon those instructions such as resource constraints, environment checks, etc.**
  - Runtime shall support basic use cases described on slide five
  - Runtime shall be capable of acting upon the basic instruction set
  - Runtime shall perform at no less than 10-20% slower than native technology
  - Runtime shall take into account security requirements such as ACL, remote installation
  - Runtime may provide a remote interface over web services based protocols
- **SDD Runtime Embedded Storage – stores information about change to the software such as install, maintenance, hot fix, uninstall, etc.**
  - Embedded storage may be a relational database
  - Embedded storage shall house either a critical subset of SDD data or the SDD itself
  - Embedded storage shall provide a listing service for viewing of content
  - Embedded storage shall provide a service or utility for direct editing of content
  - SDD Runtime may provide a remote interface over web services based protocols
- **SDD Service – serves as a local or remote reference to a runtime**
  - Service shall expose availability/capability of runtime(s)
  - Service shall keep an installation index up to date for easy lookup of deployed software
  - Service may provide a remote interface over web services based protocols
- **SDD Installation Index – an index for lookup of an SDD service**

Why Open Source?

- **Open source will**
  - Promote and accelerate the adoption of SDD
  - Provide a reference implementation for the OASIS SDD standard that can enhance interoperability
  - Provide a base runtime that commercial vendors can extend
  - Allow collaboration to solve deployment problems which will help with issues such as integration, compatibility, etc.

- **SDD does *not* define runtime behavior**
  - OASIS Technical community desires to work together on construction of a runtime

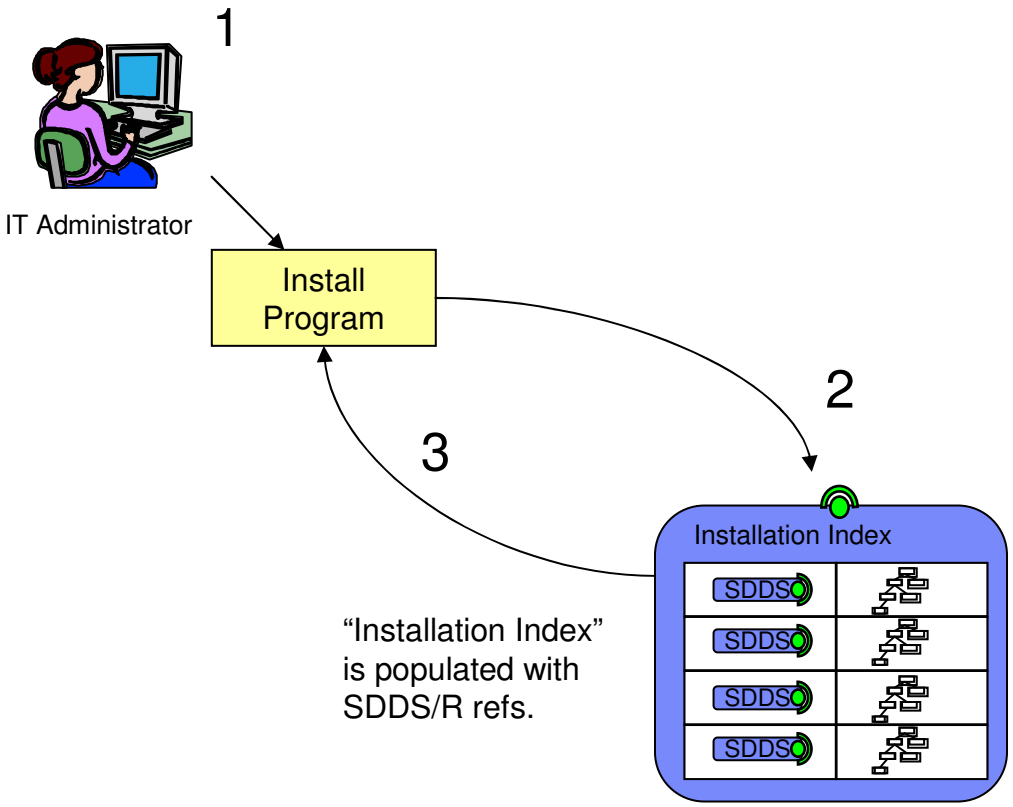
- **A reference implementation in required for Oasis approval of the specification**
  - An open runtime will facilitate the creation of reference implementations

Community Value Derived From Open Source

- Benefit to open community
  - Uses standards throughout
    - SDD, CIM/SML/CML, Web Services
  - Lower Total Cost of Ownership
    - Development expense for “infrastructure” shared among community
  - Higher quality software, e.g. collaboration between vendors brings in best ideas of each
  - Demonstrated value through exemplary applications
- Benefit to commercial vendors
  - Standards-based software lifecycle management
    - Consistency across vendors in describing deployment artifacts & expressing dependencies
  - Reference implementation is pubicly available to enhance building/deploying of complex composite applications
  - Easily extended and built upon by commercial vendors
  - Enables applications to be ready for deployment by participating provisioning systems

Example Scenario

Use Case “Install to Production”: Step 1

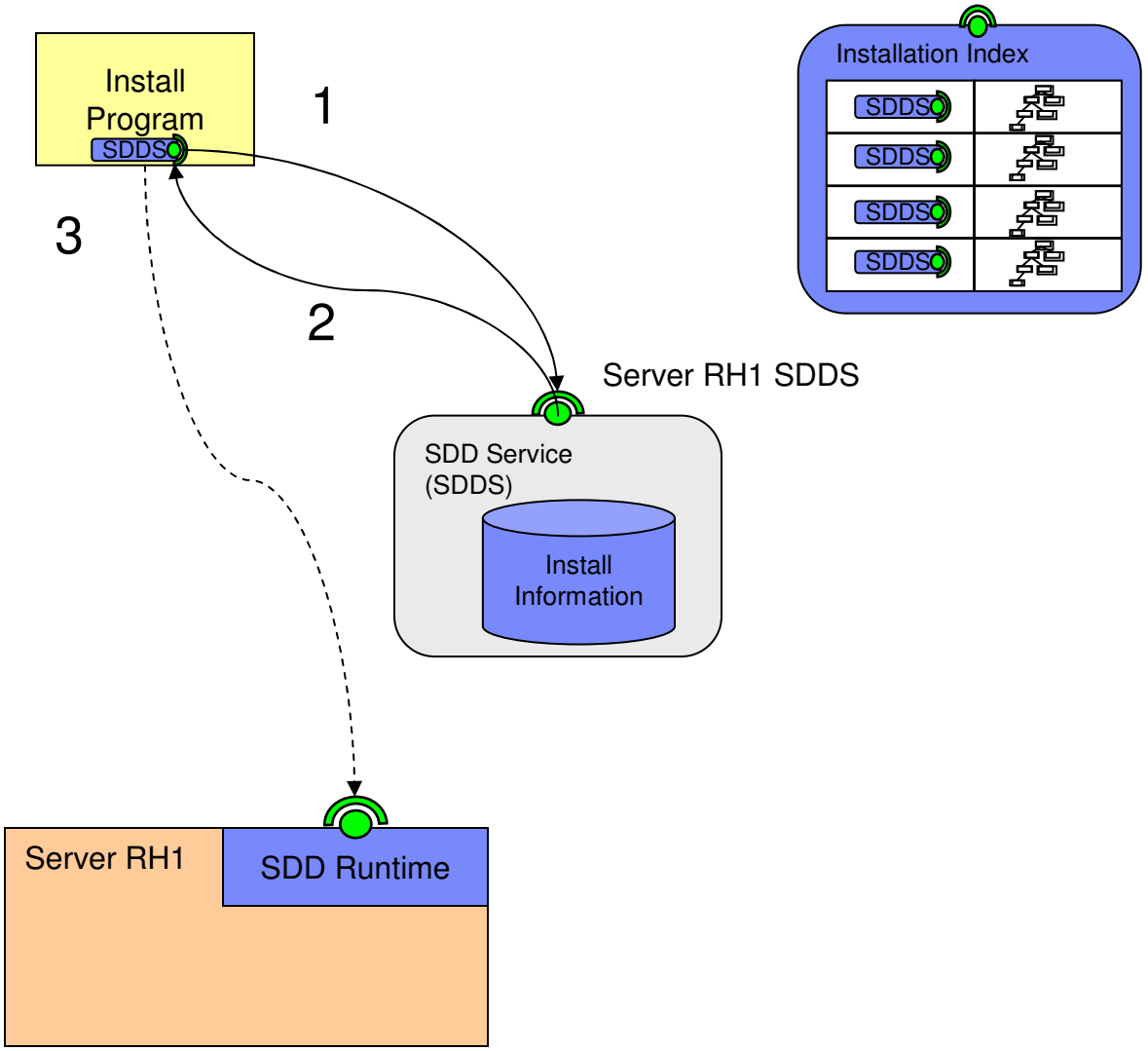


1. The IT Administrator wants to install version 2 of the Accounting application on Server RH1. This will replace version 1 of the same application

2. The install program needs to access the SDD Service on RH1 to make the change. It queries the “Installation Index” for the SDD Service (SDDS) that contains the information for Server RH1. **An alternate query could be for all servers that have version 1 of the Accounting application.

3. A reference to the SDDS is returned. If the deployment is distributed, this would be an EPR.

Use Case “Install to Production”: Step 2



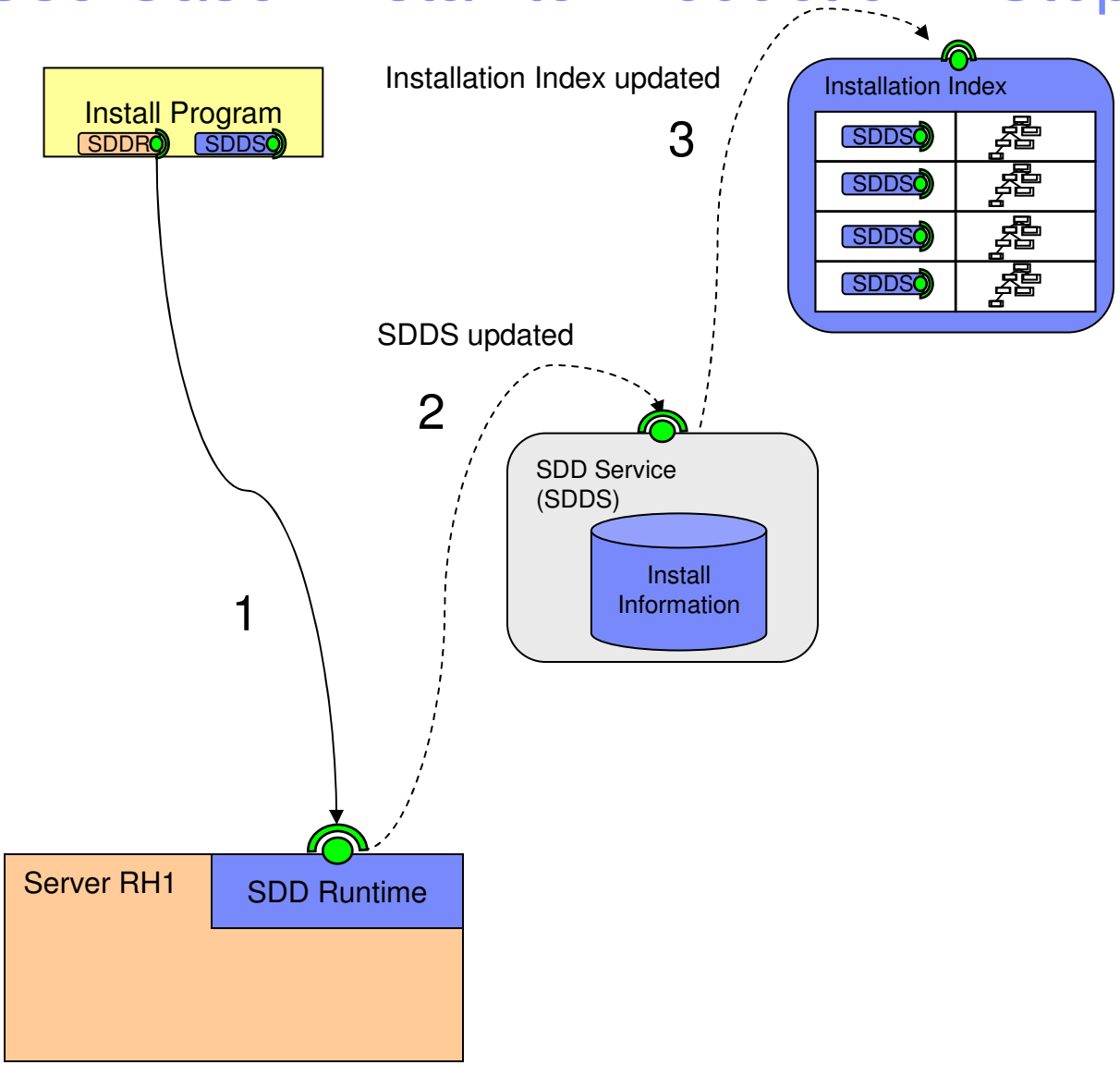
The SDD Service contains the management data of the installations.

1. The install program can retrieve information about the capabilities of the service by using WS-Metadata Exchange (WS-Mex)

2. The client could ask the SDDS for the current set of installed software on the Server RH1

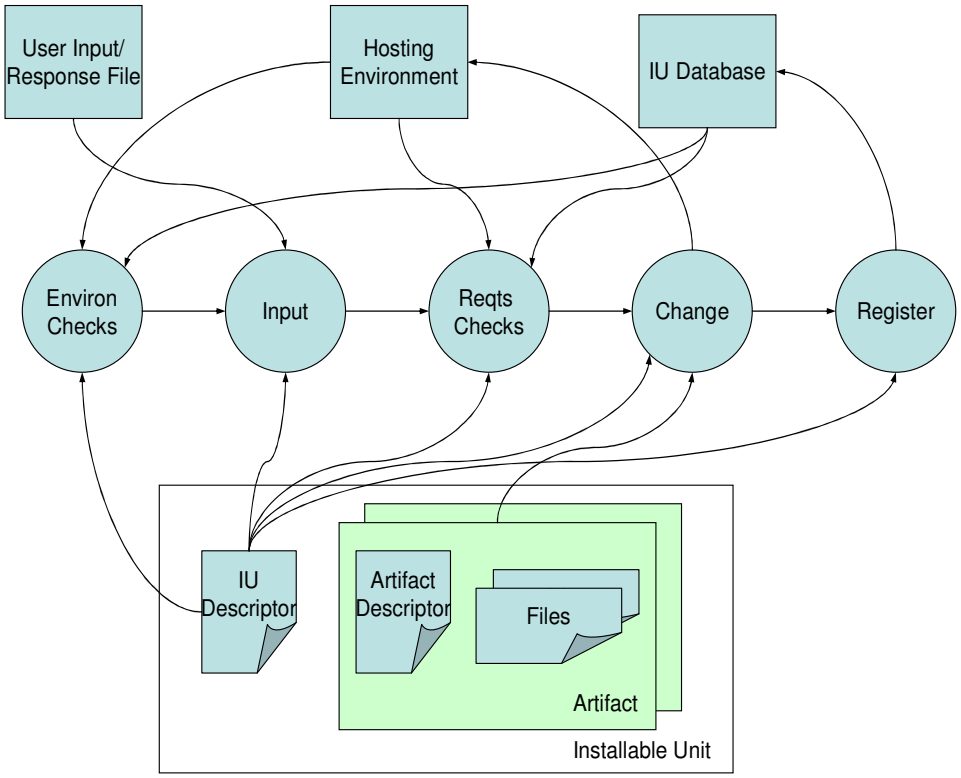
3. An additional piece of information that may be exchanged is the reference to the SDD runtime (SDDR) that can perform an install. This may be the same as the SDDS reference (local machine instance) or it could be distinct (CMDB use case)

Use Case “Install to Production”: Step 3



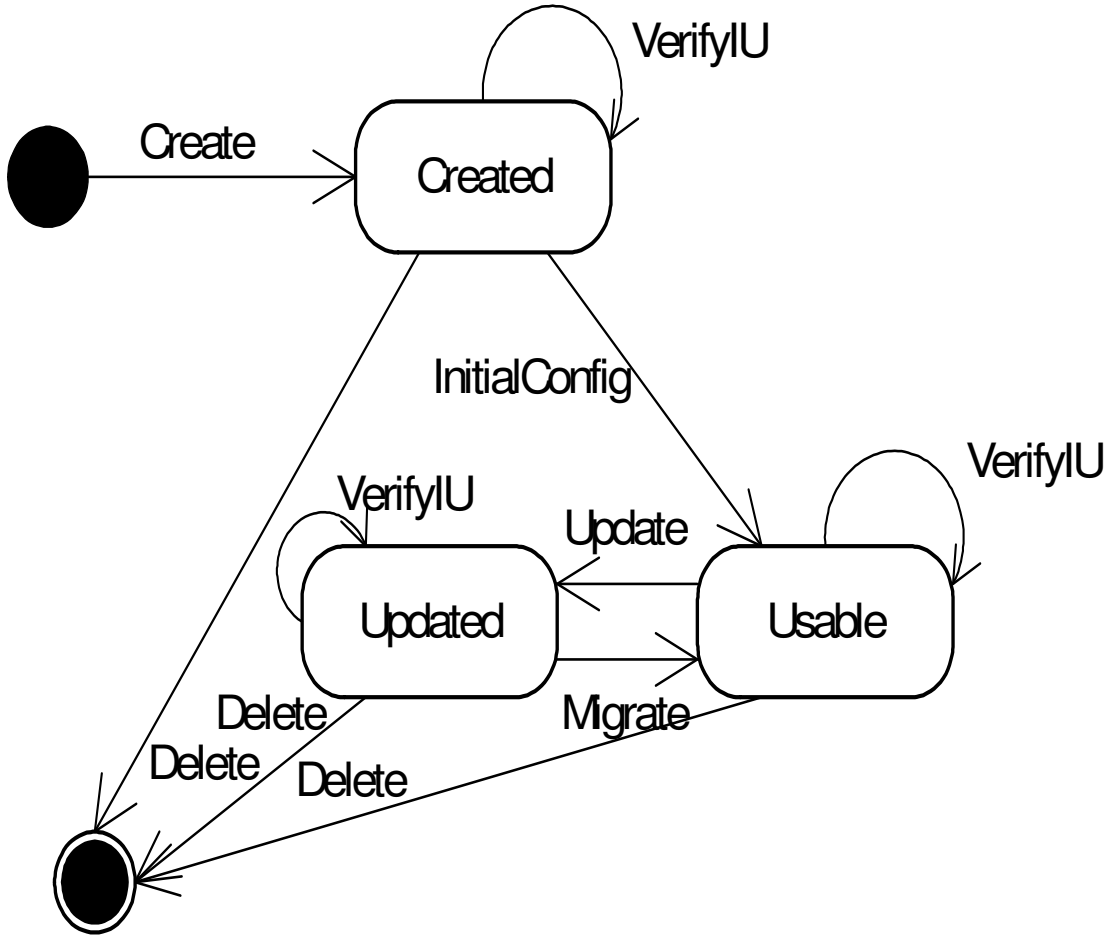
1. Given a reference to the SDD runtime, the install program can orchestrate the deployment.
2. Following a successful installation, the install information contained in the SDDS will need to be updated. This can happen via an event, e.g. a WS-Notification. This would facilitate a loose coupling of the components.
3. When the SDD Service is updated, the Installation Index will need to be updated as well with new information. This information will be the updated metadata indicating it now has version 2 of the Accounting app.

Step 3, Runtime actions details (taken from IUDD spec)



- **The runtime takes user input (via the client or some other mechanism – i.e. response file) and performs action based on the inputs**
- **Runtimes perform each of the actions represented below in the circles … environment checks, gathering of input, requirement checks, the change operation and registration of the change. One or move events may be triggered in each of these actions. Such events can be used for progress indication, orchestration, etc.**
- **Change can be new software, additional languages or changed configurations**
- **The runtime should be a framework that allows for extensions to the actions identified. The runtime executes a chain of actions which fire events. Consumers of a runtime should be able to easily add actions to the chain.**
- **Default actions should have the ability to be overridden. Consumers may replace the environment check action with their own. An action descriptor will need to be supplied to the runtime to indicate what actions to load, execution order, etc.**

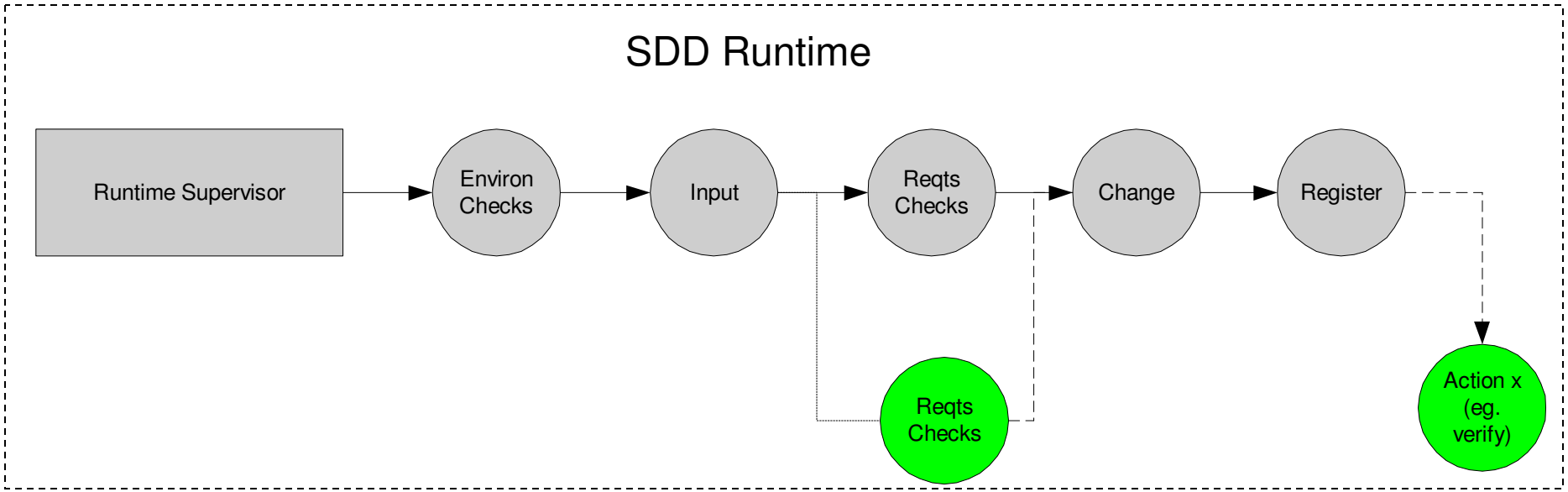
Step 3, Runtime Life cycle operations (taken from IUDD spec)



- **This diagram reflects the state of a deployed installable unit, configuration unit or localization unit (all defined in the SDD)**
- **Runtimes perform actions on artifacts based on metadata defined in the SDD. That metadata will determine artifact state in the life cycle operations associated with that artifact**

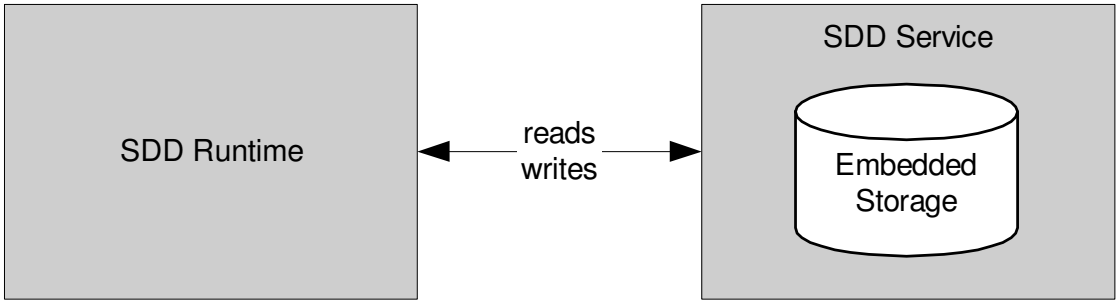
Runtime extension points

- **The runtime serves primarily as a backbone. Any of the actions that it executes can be replaced and additional actions may be inserted into the chain**
- **Below the default path is outlined in grey. Users can override actions as illustrated in the requirements checks action as well as add new actions as illustrated with the verify example. Additional actions do not have to be appended to the action chain … they can be added anywhere in the chain.**
- **This implies the runtime supervisor must follow a plugin architecture and load the appropriate modules in the appropriate order as specified by the runtime consumer. This may lend itself to an OSGi bundle/plugin model**
- **This project will NOT address the client application included in the initial slides as such the UI for driving and orchestrating these changes will be provided by the consumer of the runtime or as a commercially available product from install tool providers.**
- **A possible reference implementation would be to install COSMOS using an SDD and this runtime.**



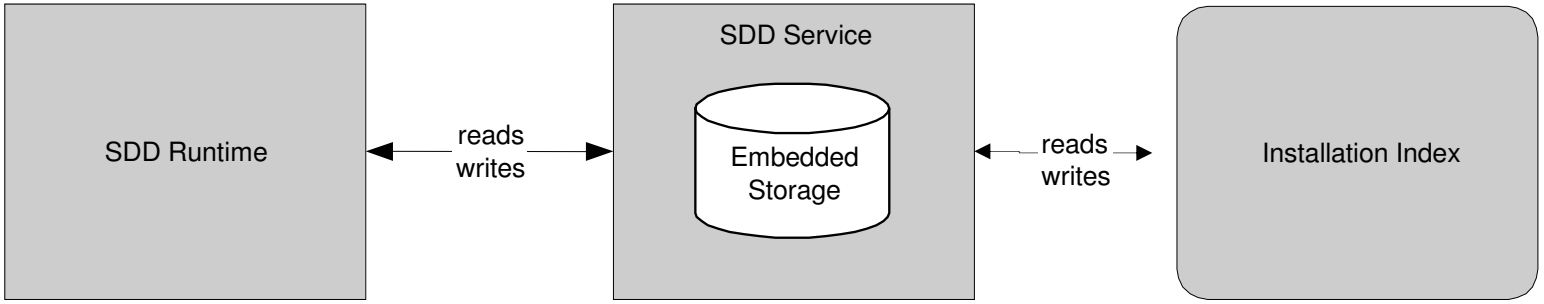
SDD Service

- **The runtime interacts with an SDD service … either local or remote. The service is used primarily for registration of the change.**
- **Registration includes software id, install location, version information, etc.**
- **Embedded in the SDD service is a data storage system … possibly Derby, although alternative data storage systems could be used.**
- **The service will need read/write APIs exposed to the runtime. The runtime will need to query the underlying installation information for software already installed as well as write data about the software it is deploying**
- **The data model for the underlying storage system will need to be designed**



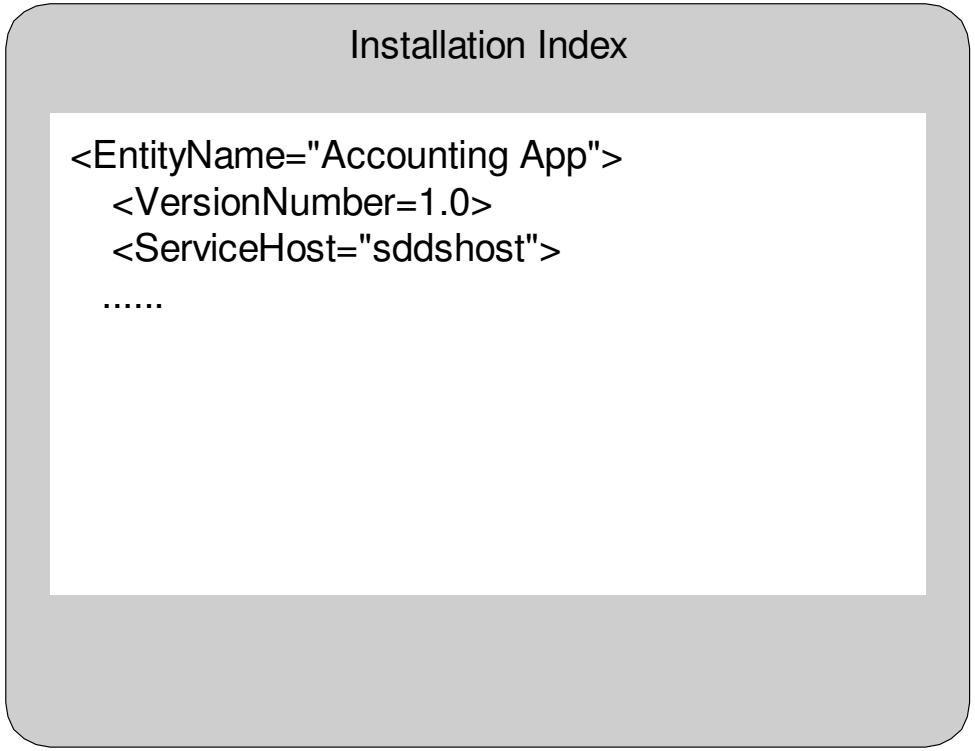
SDD Service

- **The SDD service must also update the Installation Index with appropriate information -- a handle to the installed software instance but not ALL data that goes into the underlying data storage system.**
- **Assumption is that the SDD service will require the ability to both write (update) and read (query) information in the Installation Index.**



Installation Index

- **Installation index is a relatively simple table designed to provide handles to all entities deployed in a particular domain. Assumption is that entities are all software related, but may or my not be applications … device drivers for example.**
- **The index should provide an address to the SDDS which "hosts" the install data (in the relational database mentioned earlier). All data needed to create the address is unknown at this point.**
- **May use SDMX standard for this**



Questions??

Definitions & Concepts…

Installation Index	A catalog (repository) used to find the endpoints of an SDD Runtime.
SDD	Solution Deployment Descriptor. An XML representation of a piece of software: its identity, requirements, executable, possible targets
SDD Runtime	Software that interprets the SDD package and deploys the package based on the SDD contents
SDD Service	A runtime construct that exposes a well defined API for accessing the installation information of a set of resources.
IU	Installable Unit: the basic unit/component of a software application. Can be installed standalone or with another component.
Artifact	The actual files to be deployed