CBDIJournal

Product Report

Software AG CentraSite ActiveSOA

CentraSite is a SOA governance and lifecycle management product from Software AG. This report looks at how an enterprise user organization may customize and extend the provided capabilities to support the meta model underlying Service Lifecycle Automation and SOA Configuration Management as discussed in recent reports.

By Lawrence Wilkes

Originally published in the April 2009 edition of the CBDI Journal

CBDI

Independent Guidance *for*

Service Architecture and Engineering

Product Report: Software AG CentraSite ActiveSOA

CentraSite is a SOA governance and lifecycle management product from Software AG. This report looks at how an enterprise user organization may customize and extend the provided capabilities to support the meta model underlying Service Lifecycle Automation and SOA Configuration Management as discussed in recent reports.

By Lawrence Wilkes

Introduction

Centrasite ActiveSOA is the new product name for Software AG's merged portfolio of SOA governance and asset management products. It combines:

- the metadata repository and asset management capabilities of CentraSite Enterprise.
- together with the policy enforcement and governance capabilities of CentraSite Governance Edition. Which itself was a rebranding of the webMethods Infravio X-Registry that Software AG acquired in 2007.

As such, it now provides a comprehensive set of capabilities to support the service lifecycle.

General availability of CentraSite ActiveSOA is scheduled for Q2 of 2009. A Community Edition is also scheduled.

Content Structure and Meta Model

It is very difficult to set and enforce policies where the information is unstructured. A well-defined meta model is therefore at the core of enabling SOA governance. Without the appropriate level of precision and granularity in terms of how items are defined and stored in an asset management system it is impossible to apply any rigorous governance in an automated or semi-automated fashion.

ActiveSOA provides a meta model consisting of the following object types.

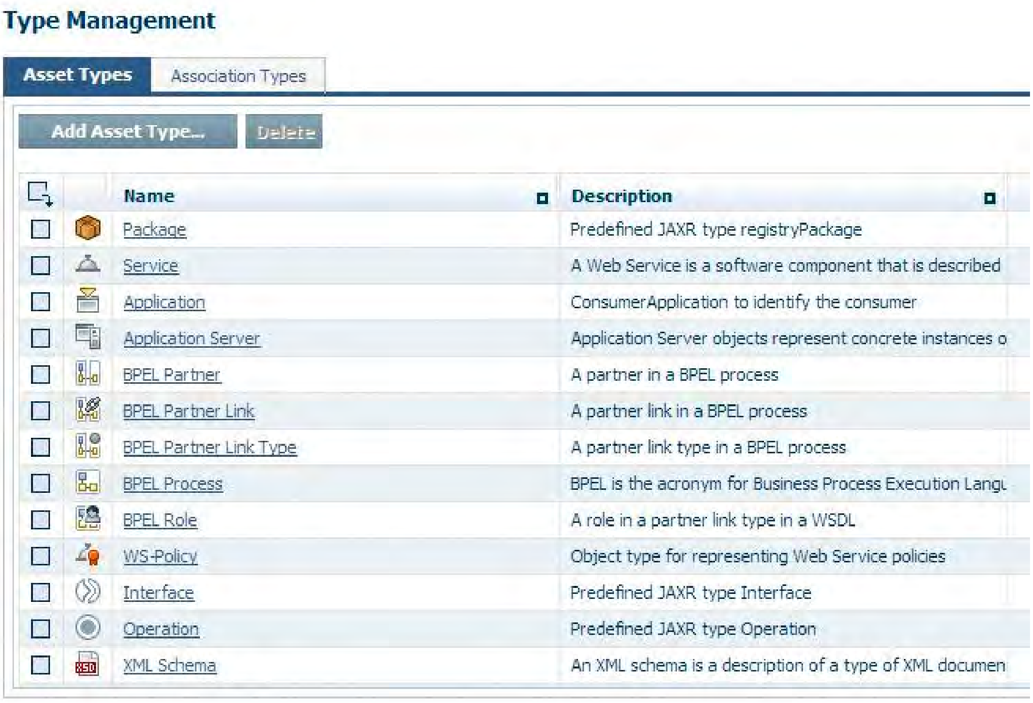
- Organizations
- Users
- Taxonomies
- Policies
- Assets
- Report Templates
- Lifecycle Models

ActiveSOA ships with a relatively small set of pre-defined Asset Types including the 'usual suspects' such as:

- Service (Web Service) and Service Binding
- Interface and Operation Package
- Consumer Application

- Application Server
- Various BPEL assets
- XML schema

Some of these can be seen in the screen capture in Figure 1.



□		Name	Description
□		Package	Predefined JAXR type registryPackage
□		Service	A Web Service is a software component that is described
□		Application	ConsumerApplication to identify the consumer
□		Application Server	Application Server objects represent concrete instances o
□		BPEL Partner	A partner in a BPEL process
□		BPEL Partner Link	A partner link in a BPEL process
□		BPEL Partner Link Type	A partner link type in a BPEL process
□		BPEL Process	BPEL is the acronym for Business Process Execution Langu
□		BPEL Role	A role in a partner link type in a WSDL
□		WS-Policy	Object type for representing Web Service policies
□		Interface	Predefined JAXR type Interface
□		Operation	Predefined JAXR type Operation
□		XML Schema	An XML schema is a description of a type of XML documen

Figure 1 – Some Pre-defined ActiveSOA Asset Types (source: Software AG)

A UI wizard makes it straightforward for users to extend the meta model. Users can define their own custom Asset Types, as well as defining their properties and attributes.

Custom attribute extension of the meta models is typed, which enables for example the use of date pickers and other built-in UI mechanisms for searching assets or applying governance rules. For example, policies can automatically be applied to assets with an attribute of "creation date" that falls between two arbitrary dates.

Associations

Association Types can also be defined to show types of relationships between different Asset Types. Once Association Types have been defined, these can then be referenced in a Relationship Attribute of an Asset Type. For example, the association '*part of*' is widely used in the CBDI-SAE meta model[1]. By creating this as an Association Type it can then be added as a Relationship Attribute to '*Business Type*' (an Asset Type), to show that it is '*part of*' a '*Business Domain*' (another Asset Type).

Classifications

Classification Attributes enable an Asset Type to be associated with a taxonomy. For example an organization may wish to classify Services according to CBDI-SAE Service Architecture Layers[2].

ActiveSOA only ships with a limited number of taxonomies. These are Asset Type, Organization, and the North American Industry Classification System and Geographical

Location codes, both of which come from UDDI. However, new taxonomies are also straightforward to create via the UI.

Profiles

Profiles can also be defined to create multiple, different ‘views’ of an Asset Type for presentation. For example various default profiles are provided with ActiveSOA that show views such as Summary, Permission, Version, Classification, Association, and Consumer information.

Supporting Documents

ActiveSOA also provides a repository to store additional documents that can be associated with an asset. A File Attribute can be added to an Asset Type to hold a reference to those documents. If the supporting document is held elsewhere however, then a simple URL type attribute can be used to hold its address.

This is most frequently used for documentation, and policies can be established and enforced for ensuring that documentation meets the requirements of the service lifecycle as well.

Extension

It certainly is straightforward to extend the ActiveSOA meta model through the UI. However, the danger is that it may almost be too easy, encouraging users to grow the model in an unplanned way with the risk of a chaotic result. As such, we advise that users should take time to properly model their requirements before commencing any extension activity. The CBDI-SAE Meta Model would of course be a good starting point! Sensibly, the ability to customize the meta model is controlled through permissions, and so granting users Administration permissions that would allow this should therefore be carefully managed.

Using the Asset Catalog

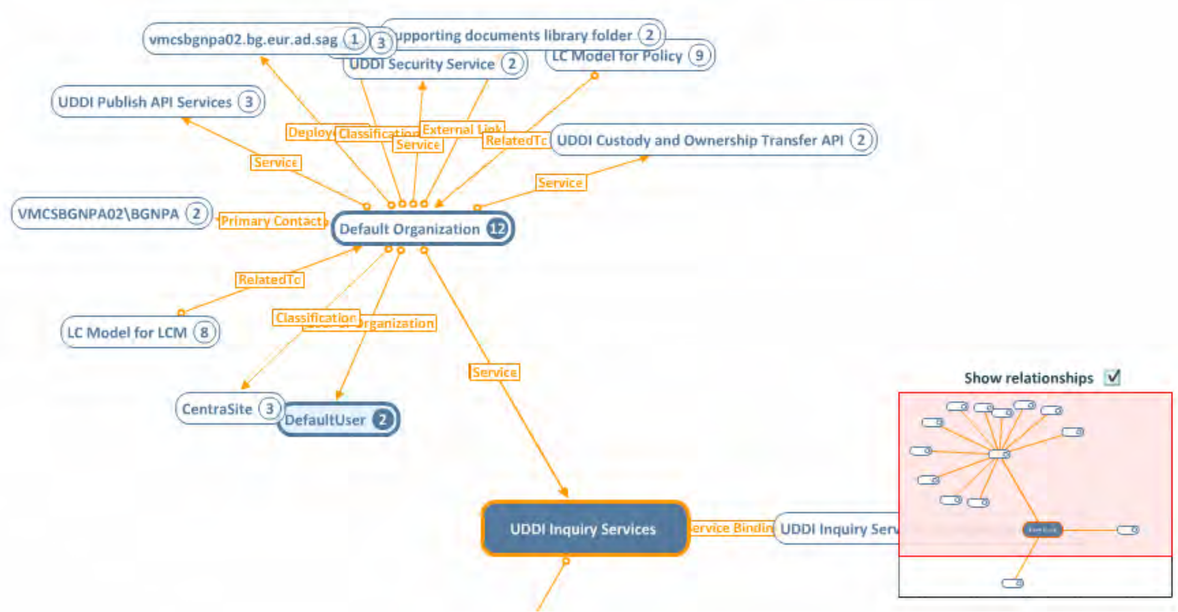


Figure 2 - Graphical View of ActiveSOA Content (source: Software AG)

Access to the Asset Catalog is controlled through a combination of roles such as ‘Asset Provider’ and ‘Asset Consumer’, and permissions such as ‘Create Assets’ and ‘Use Administration UI’.

Using taxonomies and relationships make it easy to browse and search the catalog. Relationships can be viewed graphically, as shown in Figure 2, which also gives an indication of the number of instances in a relationship. As well as improving the understanding of relationships between assets, this is useful when performing impact analysis.

Users can limit browsing and searching by classification for example, and they can also save searches so that they can be used by others. This is useful in the graphical view as dependencies can become exceedingly complex, and saved searches are essentially a way of “slicing” the dependency “space” visually.

Users can also set up event notifications to inform them when objects are read, created, updated or deleted.

Assets can be created from scratch or updated through the UI, or many cases might be imported from some external source, such as Web Services from a UDDI registry. This is further discussed later in the report.

Service Lifecycle and Governance

ActiveSOA provides a state machine to define lifecycles that can be applied to various objects including Asset Types, Organizations, Taxonomies, policies and lifecycle models themselves. Policies can then be applied to state changes and other events to support the governance process.

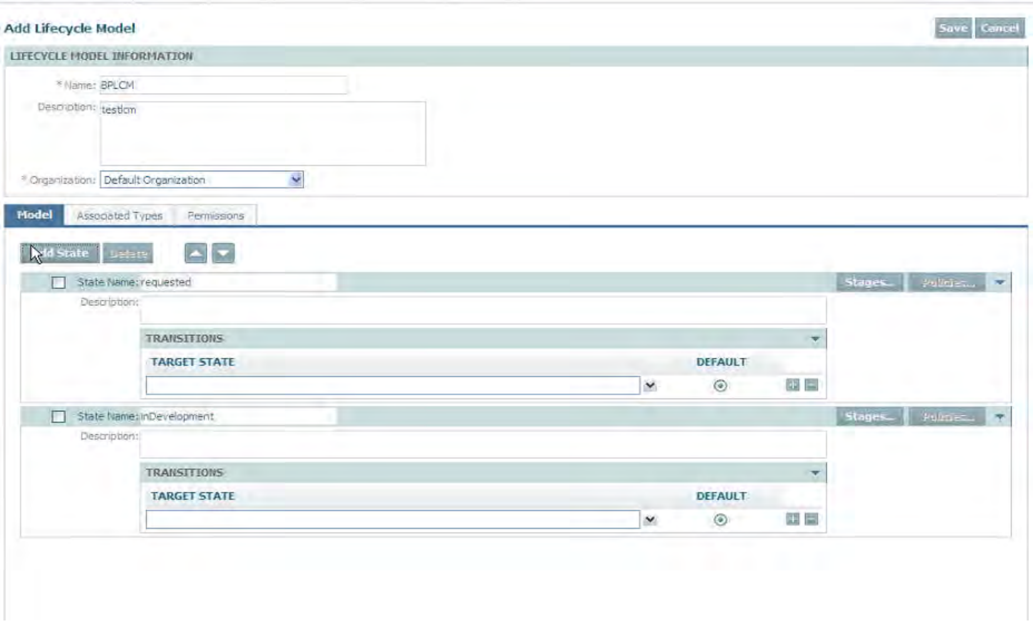
ActiveSOA allows multiple lifecycles to be defined and applied to different objects. It also allows for both ‘system-wide’ and ‘organization specific’ lifecycles, so that Asset Types for example that are owned by an organization unit can be subject to a lifecycle that is also owned by, and specific to that unit.

Users with appropriate permissions can extend and define their own lifecycle models through the UI, defining the states and transitions, the object it applies to, as shown in Figure 3.

Policies are executed on events, such as state transitions defined in a lifecycle model, or pre and post events when an object is read, created, changed or deleted. These dynamic event triggers are key to establishing complex workflows during the change time governance of highly interdependent asset types.

Policies can be set to validate the values of attributes or the existence of relationships. This can be used to enforce the constraints that may be inherent in the meta model and delivery approach that the organization is following. For example, using the CBDI-SAE meta model you would ensure that a Business Service is associated with a Business Service Subject, and that a Service Specification is assigned to an appropriate Architecture Layer.

ActiveSOA provides a set of predefined design/change time ‘Action Templates’ that form building blocks that be used to define a policy, some examples of which are shown in Figure 4. Not only can these validate attribute values, but can also set them. They can also initiate workflows such as approvals, and presume everything is approved and OK, can then set the appropriate lifecycle state.



* Name: BPLCM

Description: testlcm

* Organization: Default Organization

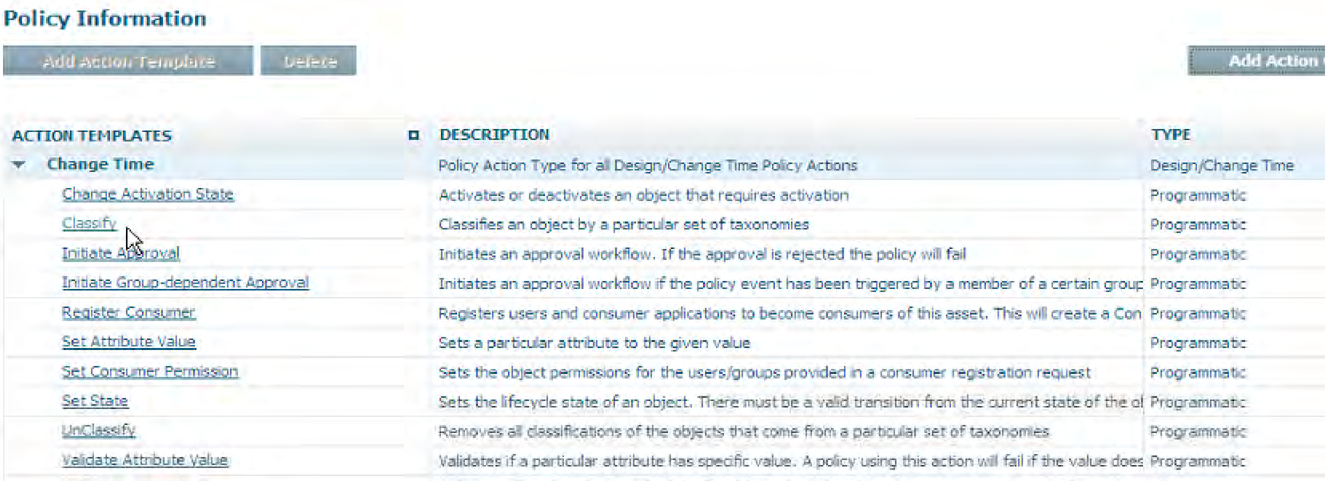
☐ State Name: requested

Description:

☐ State Name: inDevelopment

Description:

Figure 3 – Defining the Lifecycle Model in ActiveSOA (source: Software AG)

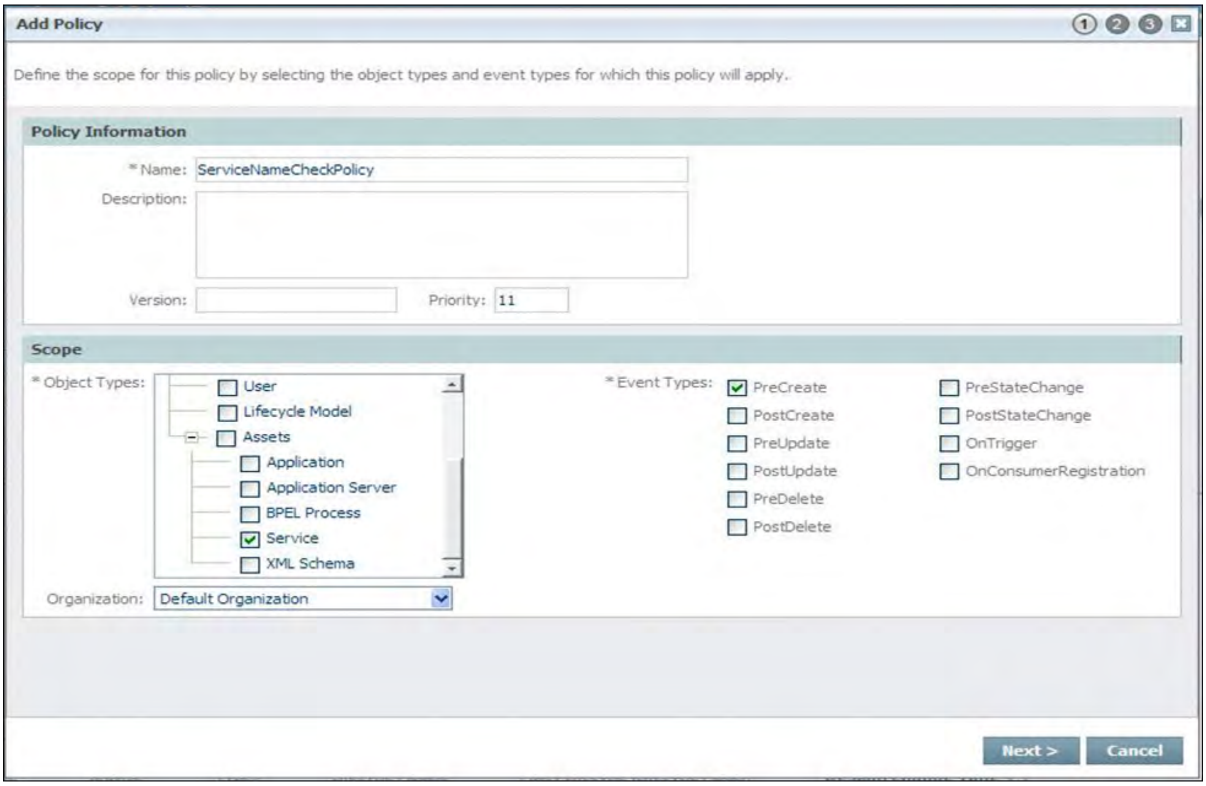


ACTION TEMPLATES	DESCRIPTION	TYPE
Change Time	Policy Action Type for all Design/Change Time Policy Actions	Design/Change Time
Change Activation State	Activates or deactivates an object that requires activation	Programmatic
Classify	Classifies an object by a particular set of taxonomies	Programmatic
Initiate Approval	Initiates an approval workflow. If the approval is rejected the policy will fail	Programmatic
Initiate Group-dependent Approval	Initiates an approval workflow if the policy event has been triggered by a member of a certain group	Programmatic
Register Consumer	Registers users and consumer applications to become consumers of this asset. This will create a Con	Programmatic
Set Attribute Value	Sets a particular attribute to the given value	Programmatic
Set Consumer Permission	Sets the object permissions for the users/groups provided in a consumer registration request	Programmatic
Set State	Sets the lifecycle state of an object. There must be a valid transition from the current state of the ol	Programmatic
UnClassify	Removes all classifications of the objects that come from a particular set of taxonomies	Programmatic
Validate Attribute Value	Validates if a particular attribute has specific value. A policy using this action will fail if the value does	Programmatic

Figure 4 - Action Templates (source: Software AG)

As their name suggests, pre-defined WS-I compliance actions can test for compliance with certain Web Service protocol interoperability profiles from the WS-I organization[3]

Policies are straightforward to author via the UI, as illustrated in Figure 5. Relevant objects and events are selected, and then appropriate actions can be defined.



Define the scope for this policy by selecting the object types and event types for which this policy will apply.

Policy Information

* Name: ServiceNameCheckPolicy
Description:
Version:
Priority: 11

Scope

* Object Types:
- □ User
- □ Lifecycle Model
- □ Assets
  - □ Application
  - □ Application Server
  - □ BPEL Process
  - ☑ Service
  - □ XML Schema

Organization: Default Organization

* Event Types:
- ☑ PreCreate
- □ PostCreate
- □ PreUpdate
- □ PostUpdate
- □ PreDelete
- □ PostDelete
- □ PreStateChange
- □ PostStateChange
- □ OnTrigger
- □ OnConsumerRegistration

Figure 5 - Authoring Policies with ActiveSOA (source: Software AG)

Users can also develop their own custom Action Templates. ActiveSOA supports both:

- manual actions where user interaction is required, such as an Approval process.
- programmatic actions that trigger action rules, typically coded as Java classes.

Using the API's described later, it is also possible to apply the lifecycle management capabilities of ActiveSOA to assets stored externally.

Policy Management

As well as design/change time policies, certain run-time policies can also be authored and managed in ActiveSOA, and then pushed out to an appropriate 'policy enforcement point' (PEP) in a run-time engine. ActiveSOA ships with built-in Action Templates for:

- Security
- Performance Monitoring
- Event Logging
- Message Validation

However, these are designed specifically to use Software AG's webMethods Mediator product as the PEP, which is designed to be configured using ActiveSOA.

In addition, many of the popular run-time PEPs in alternative products have been pre-integrated through the Centrasite Community. Integrations typically go beyond just pushing runtime policy into runtime PEPs and often also provide for runtime metrics to flow back into Centrasite ActiveSOA in order for users of the Registry Repository to understand runtime behavior of services before binding.

Users also have the ability to author their own custom actions if support for the PEP they are targeting is not already available.

The run-time policies are negotiated as contracts - unique statements of the run-time policy requirements between provider and consumer. Although the contracts are enforced by the PEP in the run-time system, they must also be negotiated by the service consumer with ActiveSOA when they decide to use the Service. This is a very useful facet of the full lifecycle capabilities of ActiveSOA.

Versioning and Configurations

ActiveSOA provides for major or minor versioning of objects. A major version creates a new 'cloned' object with its own unique id, whereas a minor version is just an update of the existing object. It is also possible to create associations between major versions of objects – as they are each just unique objects - which may be useful when multiple versions of a Web Service for example are all still operational.

There is no specific mechanism to define 'configurations' of Asset Types or other objects, along the lines of those discussed in previous reports, such as a Service Portfolio Plan consisting of various architecture models, a list of Services, Service Specifications and so forth. However, it would be straightforward to define Service Portfolio Plan as a new Asset Type together with the required Association Types and relationship attributes needed. It could also link to any supporting documents, or objects held elsewhere as discussed earlier. Policies could then check the completeness of the plan and associated assets as the plan moves through its lifecycle.

Interoperability

Most activities that can be performed via the UI can also be performed via one of the APIs provided as part of the SOALink framework, some of which are summarized in Table 1.

The SOALink Cookbook[4] describes a number of design and run-time integration scenarios, and provides guidance and some already developed APIs to integrate with other products, or use their agents.

Using the registry API, ActiveSOA can be deployed as a federated registry consisting of both ActiveSOA and other UDDI registries, providing automatic and manual synchronization. Custom plugins into the federation framework also allow for federation with arbitrary non UDDI compliant systems.

ActiveSOA can also act as a 'system of record' by providing a centralized information store into which other design and run-time tools can publish relevant information or events. For example, development tools, configuration management tools, or other registries might publish asset dependencies into ActiveSOA. Whilst systems management tools or an Enterprise Service Bus might publish metrics or run-time events to ActiveSOA.

IDE and other development tools	Eclipse plug in – can use Eclipse Registry Browser Has own interfaces into some 3rd party systems Partners or users can develop own custom imports, using guidance found in SOA Link
UDDI Registry	Can interface to other UDDI-compliant registries using • UDDI 3.0 Registry access • JAXR (Java API for Registries) supporting both UDDI 2.0 and ebXML 2.0 registries
CMDB	No specific integration to a CMDB is provided, but using the APIs, it is possible to link between objects in a CMDB and assets in ActiveSOA providing the type of object is common to both.
Other APIs	• Lifecycle Management API provides Javadoc access to lifecycle models. It is possible to apply lifecycle management to external assets, not just those in the ActiveSOA repository. • CentraSite Application Framework (CSAF) provides a programming model for developing custom extensions on top of CentraSite. • Java Management Interface

Table 1 - Interoperability Features of ActiveSOA

Representing CBDI-SAE

There are no specific considerations for defining types from the CBDI-SAE meta model or SAE deliverables. As Table 2 shows, most would be defined as Asset Types together with appropriate relationship attributes to other relevant Asset Types.

As such it would be fairly straightforward to use ActiveSOA to support a CBDI-SAE approach.

SAE Deliverable or Concept	Instantiation in Active SOA
Service (notional)	Asset Type. Relationship Attributes to other asset types such as, • One or more Service Specifications
Service Portfolio	Asset Type. Relationship Attributes to other relevant Asset Types that would be found in the portfolio plan, such as • Business Models (URL pointer to) • Service Architectures • Service (notional)

Service Architecture	Asset Type. Relationship Attributes to Service Specifications URL Attributes to point to Architecture Model , or File Attributes if the model is held in the repository as a supporting document
Service Specification	Asset Type. Relationship to other asset types such as, • Specified Operation URL Attributes to point to Service Information Model, or File Attributes if the model is held in the repository as a supporting document
Specified Operation	Asset Type. This would be added as a new asset type rather than customizing the provided Operation asset type, which would be left as is to represent Java operations Relationship Attribute to XML Schema
Service Level Agreement	Asset Type. Run-time SLAs can be defined in ActiveSOA that run in Mediator.
Automation Unit Specification	Asset Type. File Attributes or URL links to associated models and other objects used in implementation
Policy Type	ActiveSOA doesn't support the concept of a Policy Type, but you can develop custom action templates that can be reused when defining policies
Policy	Policies

Table 2 - Defining CBDI-SAE in ActiveSOA

Summary

Centrasite ActiveSOA represents an amalgamation of mature products that together have over 200 customers.

ActiveSOA is a straightforward product to use and customize via its UI. At the same time, a complete set of APIs enable it to be integrated with other products to support the SOA delivery and operational lifecycles, and there is an active community of users and partners developing 3rd party support.

Out of the box, ActiveSOA does not provide a particularly rich set of pre-defined asset types, and is focused on implementation and run-time assets, but this is no different to similar products. However, the list of asset types is easily extended. Similarly it is easy to create other objects such as taxonomies, lifecycle models and policies.

We can imagine that organizations would be 'up and running' quickly with ActiveSOA. However, as mentioned earlier we still recommend that they carefully plan their extensions to the meta model, as well as deciding roles and access control models first.

Links

Centrasite http://www.softwareag.com/corporate/products/wm/soa_governance/default.asp

Centrasite Community for plug-ins
http://communities.softwareag.com/ecosystem/communities/public/developer/centrasite/

[1] CBDI-SAE Meta Model V2.0.

[2] See Appendix A1. Architecture Layers, in CBDI-SAE Meta Model V2.0.

3 http://www.ws-i.org/deliverables/Default.aspx

[4] SOALink Cookbook. http://documentation.softwareag.com/webmethods/soal8/overview.htm

About CBDI

CBDI Forum is the Everware-CBDI research capability and portal providing independent guidance on best practice in service oriented architecture and application modernization.

Working with F5000 enterprises and governments the CBDI Research Team is progressively developing structured methodology and reference architecture for all aspects of SOA.

CBDI Products

The CBDI Journal is freely available to registered members. Published quarterly, it provides in-depth treatment of key practice issues for all roles and disciplines involved in planning, architecting, managing and delivering business solutions.

Visit www.cbdiforum.com to register.

Platinum subscription – A CBDI Forum subscription provides an enterprise or individual with access to the CBDI-SAE Knowledgebase for SOA delivering ongoing practice research, guidance materials, eLearning, tools, templates and other resources.

Everware-CBDI Services

At Everware-CBDI we enable large enterprises and governments to become more agile by modernizing their business systems. We have repeatable processes, resources, tools and knowledge-based products that enable enterprises to transform their current applications in an efficient, low risk manner, into an optimized service-based solutions portfolio that supports continuous, rapid and low cost evolution. Our consulting services range from providing practices and independent governance to architecture development, solution delivery and service engineering.

Contact

To find out more, and to discuss your requirements visit www.everware-cbdi.com or call

USA and Americas: 703-246-0000 or 888-383-7927 (USA)

Europe, Middle East, Africa, Asia, and Australasia: Telephone: +353 (0)28 38073 (Ireland)

www.everware-cbdi.com

IMPORTANT NOTICE: The information available in CBDI publications and services, irrespective of delivery channel or media is given in good faith and is believed to be reliable. Everware-CBDI Inc. expressly excludes any representation or warranty (express or implied) about the suitability of materials for any particular purpose and excludes to the fullest extent possible any liability in contract, tort or howsoever for implementation of, or reliance upon, the information provided. All trademarks and copyrights are recognized and acknowledged. The CBDI Journal may be distributed internally within customer enterprises that have current corporate subscriptions. Otherwise CBDI Journals may not be copied or distributed without written permission from Everware-CBDI.