Lifecycle Manager 2019.1.20

Concepts and Facilities Guide

Lifecycle Manager

Concepts and Facilities Guide
Version 2019.1.20
August 2020

Copyright

Trademarks

Perforce Software

Perforce Software
400 First Avenue North #200
Minneapolis, MN 55401
612.517.2100
www.perforce.com
info@perforce.com

Disclaimer

Contents

1 Using this Guide

The goal of this document is to introduce the basic concepts of Lifecycle Manager and provide enough background information to use Lifecycle Manager effectively. The document's primary audience is the team responsible for the implementation and support of the Lifecycle Manager Asset Library. Additionally, asset users, asset publishers and owners, and the personnel controlling the use of the Library on an on-going basis may also find it useful.

1.1 This document and other resources

This guide is designed to facilitate understanding of how the system might apply to a particular environment. It is not intended to provide the information required to determine or initiate a specific implementation. For implementation and usage assistance, Akana provides:

- **Governance API documentation.** Lifecycle Manager provides a fully configurable process automation capability accessible via a set of REST APIs. These APIs are designed to easily integrate with Alpaca-based clients and are documented at http://docs.akana.com/lm/assets/Governance_API.pdf
- **AnySource™ Asset Adapter documentation.** These documents introduce the Asset Adapter toolkit and describe how to configure and extend it to integrate Lifecycle Manager with your own IT development infrastructure. The AnySource Asset Adapter user guide is available at http://docs.akana.com/lm/assets/AnySource_Asset_Adapter_User_Guide.pdf.
- **Library Configuration Guide.** This document provides extensive documentation on various validation and process automation options available to customers. It also provides configuration examples to help customers get started in applying their architectural and project management governance processes to automated Lifecycle Manager deployments.
- **Integration Guides.** Various documents are available at http://docs.akana.com/lm to guide administrators in the process of establishing integrations with other Akana products.
- **Training.** For a comprehensive education on Lifecycle Manager, Akana offers training for asset users and the team responsible for implementing and managing the Library and its assets.
- **Online Help.** For detailed user guidance and documentation, Lifecycle Manager includes a built-in comprehensive online help system.
- **SI Partner Community.** Our trained SI Partner Community can provide an array of services to meet your needs. Please refer to the partners section on http://www.akana.com.
- **Professional services.** Our Professional Services staff is expert at working with customers to understand and help build a plan for the most effective use of Lifecycle Manager that will meet an organization's objectives, requirements, and IT environment. Each engagement builds from a set of tried and true baseline configurations and supporting documents that are continually being developed and refined by the Akana team.
- **Online Support**. Visit http://support.akana.com for a comprehensive support site, access to the latest Lifecycle Manager service packs, and other resources.

2 Introduction to the Lifecycle Manager Asset Library

2.1 Why deploy a development-time asset repository?

Technology innovations such as Service Oriented Architecture (SOA), REST/JSON APIs, microservices, and the various runtime platforms supporting these capabilities make it possible to build applications with significantly less effort by leveraging common APIs/services and reusable components. However, many enterprises deploying these technologies are not seeing the expected benefit because the knowledge necessary to use these assets is not available and/or easily accessible. If it is difficult to search for and locate a common API, service or component, then the asset cannot be reused resulting in redundant development. If it is hard to determine the purpose of an asset and how it relates to business processes, then duplicating the work may become expedient to ensure that the application meets requirements. Even when a potentially reusable asset is found, it is often not clear if the asset has been developed with sufficient scalability, security, or other nonfunctional criteria to support its potential reuse under specific conditions, or even simply to quickly understand the full functionality of the asset due to lack of useful documentation.

To address these roadblocks, development teams are deploying asset management systems and metadata libraries (also referred to as **development-time repositories**) that store assets and knowledge about them and provide automated and role-based governance over production and consumption of those assets. Often these systems are directly associated with new development initiatives (Component-Based Development (CBD), Service-Oriented Architecture (SOA), API-Based Development, microservices, Internet of Things (IoT)). Such platforms may also be deployed as part of a broader initiative such as “enterprise architecture” or “enterprise reuse.” In either case, creating a central library of Software Development Assets (SDAs) is a critical and necessary step to deliver on the development promise of such technologies.

2.2 What is a Software Development Asset (SDA)?

SDAs consist of the various elements used or referenced in the creation or integration of software applications or services. Assets can include components (e.g., JEE, OSGi, .NET), legacy systems, services, APIs, schemas, business processes and any other executable aspect of an enterprise’s IT infrastructure as well as intellectual capital and mindshare information like architectures (both business and technical), patterns and best practices. In general, assets come in many forms, and are valuable elements that should be made available to development teams. An organization may use many different asset types, and Lifecycle Manager is designed to hold them all. SDAs can be grouped into two general types: executable assets and knowledge assets.

Examples of executable assets include:

- Components (JEE, OSGi, .NET, …)
- Frameworks (e.g., Apache Struts, Microsoft .NET Application Blocks)
- APIs and Services (SOAP-based Web services, REST/JSON APIs, message queues, …)
- Applications and their pertinent application programmable interfaces (APIs)

- Schemas (XML, JSON)
- Representations of functional capabilities and data

Examples of knowledge assets include:

- Patterns
- Best practices
- Architectures
- Reference implementations
- Business domain knowledge
- Project and development process templates

2.3 What is Lifecycle Manager?

Lifecycle Manager is a development-time Software Development Asset (SDA) repository and governance platform (i.e., library) that allows enterprises to easily represent and search for key development assets (both industry-standard and in-house developed) within an application development environment in a graphical, intuitive way. Lifecycle Manager provides an intelligent inventory of assets; their relationships to each other, to the technical infrastructure, and to the company's business architecture. Through the use of Lifecycle Manager, organizations can accelerate reuse and SOA initiatives, as well as improve the governance over production and consumption of services and other reusable assets. Application developers, business analysts and technical and business architects can search the repository for the company's SDAs, and identify those that best match business and technical requirements for application development and integration.

Lifecycle Manager can hold internally created assets, externally purchased assets, and assets that are shared with partners or consortiums.

Lifecycle Manager provides a multi-faceted index to its SDAs. Usually, some of an SDA's work products (i.e., artifacts) are sourced elsewhere with Lifecycle Manager referencing the source location(s) of these elements. These are referred to as by-reference artifacts. For example, assets may include elements kept in one or more "systems of record" outside of Lifecycle Manager, such as a document management system, a version control or source change management repository, a defect tracking system, etc. Lifecycle Manager can also directly hold artifacts such as test plans or requirements in its repository. Such artifacts are referred to as by-value artifacts. In either case these artifacts are both indexed into Lifecycle Manager's text- and XML-based search engine and made available for user review and use.

The information and references Lifecycle Manager holds for each SDA includes identifying information (Asset Name and Version) and an Asset Description and Overview. In addition, Lifecycle Manager holds various other elements which are configured to describe the specific Asset Types in the Library, including:

- **Classifiers**: such as asset type, supported platforms, functional domain & sub-domain, support level, etc.
- **Artifacts:** files or reference elements like interface definitions (e.g., WSDL or REST documentation), executable components, user documentation, test scripts, models, etc.

- **Asset Relationships**: references to other assets that represent previous versions, prerequisites, co-requisites, dependencies, etc.

2.3.1 The SDA Production/Distribution/Consumption Lifecycle

Just because you can make something into an SDA doesn't mean that you should. What makes a component, a Web service, an API or a design pattern a good SDA candidate – in other words, what makes an SDA reusable? For example, does a JEE component become an asset simply by providing its jar file? Unless the component's functionality is so dead-simple as to be totally obvious, the answer is "probably not." While the deployable jar is a very important work product (i.e., artifact) of the software development process, it in and of itself does not make the component an asset. In order for something to be considered an asset to the IT organization, it must be maintainable, discoverable, and consumable. ***Maintainability*** introduces such concepts as version control, models and other design documentation, and requirements traceability (why was the asset implemented in this way from a technical and business perspective). ***Discoverability*** forces us to consider how we help potential consumers of this asset find the asset in a timely fashion – via keywords, domain taxonomies, and mapping to models, for example. ***Consumability*** involves looking at the asset from the point of view of the downstream project planning to use the asset: is there a user guide, a well-documented API, sample client code, and other artifacts available to help the user rapidly understand how to apply this asset to the project at hand? Are dependencies to other assets (and to prior versions of this asset) specified and easily navigated?

How can you achieve these "ilities" in an SDA? By applying appropriate development processes to asset ***production***. To a large extent, these SDA production processes are simply applying a set of good SDLC processes to asset development; however, keep in mind that since you are producing an SDA for eventual library distribution and external consumption and thus your standard SDLC's artifacts may not be sufficient. For example, you may need to insist upon more complete and/or formalized documentation, and you may also want to mandate that the asset has a set of full source sample code provided alongside the executable to show examples of how the asset can be properly used. Such additional "full product" documentation and supporting work products are particularly important when you are sharing APIs with business partners both inside and outside of your enterprise (Akana provides a wide-ranging set of API-related documentation at http://docs.akana.com/cm/index.html). These types of asset production governance rules should be supported and enforced by your asset library, and Lifecycle Manager provides a wide range of capabilities to do just that. Lifecycle Manager's asset template structure allows you to establish validation rules such as specifying mandatory artifacts, providing controlled (enumerated) lists of valid classifier values, and the like. Lifecycle Manager's configurable process engine allows organizations to automate their own asset production review processes, including automatic asset validation upon submission (e.g., schema validation, policy conformance, automatic invocation of testcases) and multi-stage review steps based on the organization's own defined roles.

Ultimately, the process of building an asset results in the asset being represented by its metadata – that is, data that describes the asset from various points of view. This metadata presents a composite view of the asset across its entire development and deployment lifecycle, with indexes (i.e., references) into

the various point tools such as document management systems, requirements management systems, version control repositories, defect tracking systems, test automation tools, etc. that hold the work products or artifacts of the asset. It is this metadata that Lifecycle Manager ***distributes*** to various candidate ***consuming*** organizations (typically application development project teams) for their potential use. This distribution might be through Lifecycle Manager's native IDE integrations, or perhaps through its integration with Akana's Community Manager portal. Appropriate governance rules over consuming SDAs also comes into play – for example, a health care organization may have HIPAA-specific controls placed around access to a Web service that exposes patient information, or perhaps only specific project teams should be allowed access to a payroll management component. Lifecycle Manager enables and automates these types of consumption governance processes, allowing organizations to more effectively manage SDA usage within their development projects.

Any successful SDA reuse initiative will eventually have to deal with the issue of asset versioning. Sooner or later, your organization will reach the point where the next set of business process requirements affects one or more of your existing SDAs. How are you going to provide the necessary support for this new process while preserving a stable operating environment for existing service consumers? Because reusable SDAs are meant to be used in more than one application, organizations need to plan for the incremental enhancement of their services over a long deployment lifetime. In effect, organizations planning to build a robust, stable and extensible SOA or suite of APIs need to treat their services/APIs as "products". This includes concepts such as maintaining a well-defined and regular release cycle for reusable SDAs, maintaining backwards compatibility between SDA versions wherever possible, and providing appropriate mechanisms for new asset version requirements gathering and management. Lifecycle Manager can assist in such ongoing SDA management, allowing organizations to describe assets in any state of development (including agile/iterative development methodologies), from planned and deployed, to retired. Updates or new versions of assets are managed by the system and can be communicated to Library users.

Since Lifecycle Manager can hold such a broad range of SDAs, it is a key enabling technology for enterprise SOA transformation, integration or consolidation efforts. Having a central place to search, access and deploy assets accelerates the success of both pilot and enterprise-wide SOA initiatives.

2.4 What assets are important?

While all types of assets can be stored in Lifecycle Manager, it is not necessarily intended to hold every possible asset. The types of assets and Reference Models that are appropriate for a repository will depend on organizational objectives, available assets, asset users, and the approach implemented to locate assets. The assets recorded in Lifecycle Manager will also vary depending on the production environment and objectives.

As a development environment is integrated with Lifecycle Manager, assets are usually loaded into the system in stages, starting with those that will be most useful immediately, such as assets specifically designed for re-use. Good candidates might be:

- Services or interfaces into central applications, such as ERP and CRM systems

- Shared frameworks
- Common services
- APIs designed for external consumption
- Common technical components (e.g., logging, exception handling)
- Enterprise or Line of Business data schemas or views

Knowledge assets such as architectures, best practices, patterns, and project templates are also good candidates for initial loading into Lifecycle Manager. Such knowledge assets often represent core information that is crucial to the enterprise going forward, and Lifecycle Manager can dramatically improve the enterprise's ability to disseminate and manage such information.

When considering asset selection and staging, determine which Reference Models should be captured to show the overall architecture. Sample applications or source code snippets using the assets are also helpful.

2.5 What is an asset template?

The template capabilities of Lifecycle Manager allow users and administrators to create a comprehensive Asset Classification methodology (i.e., defining types of assets and the information held about them). Templates are used to:

- Define the information kept for each asset type in the library.
- Set valid values and information for each asset type.
- Protect the integrity of asset data through enforcing the use of valid values.

To accelerate library implementation, Lifecycle Manager ships with a default Global Asset Definition template and multiple predefined asset constraint templates. These templates are XML documents used to define and extend the Lifecycle Manager asset schema. They can be used as-is or easily configured to meet specific installation requirements. The predefined templates serve as examples to draw from, and a means to explore the Library before configuring new templates. Lifecycle Manager preloaded content also uses these predefined templates.

- **The Global Definition template** (GDT) defines the superset of element types that are available when defining constraint templates for each asset type. In other words, the GDT establishes the library's content model that is in turn further refined for each asset type to be represented in the Library.

- Among others, the following default **Asset Constraint templates** are provided with Lifecycle Manager:
    - **API:** a template suitable for provisioning and documenting APIs to the Community Manager portal. In simple terms, an API is a "productized" service providing supporting materials and capabilities such as extended user guides, legal agreements for external consumers, licensing modes exposing service operations to consumers under the scope of varying Service Level Agreements (SLAs), and so on.

- **Service:** a template suitable for SOAP- or REST-based service assets fully described by a single WSDL. RAML, Swagger or WADL document combining both interface and implementation aspects. In Lifecycle Manager terms, the Service represents the functional capability to be exposes to potential consumers.
- **Software Component:** a template suitable for component-based executable assets
- **Knowledge:** a template suitable for documenting patterns, best practices, architectures and other related knowledge asset types
- **Application:** a template suitable for applications of various types. Application assets are typically used to represent consumers of services, components and other reusable SDAs, although they can also be managed as SDAs in their own right.
- **Schema**: a template suitable for XML schema documents
- **Technical Policies and Runtime Configurations**: templates used to represent provider-side enforceable Service/API policy concepts such as security, privacy, monitoring and QoS, and collections of these policies assembled to establish supported consumer usage patterns for Services/APIs
- **Community Manager Tenant**: a template used to represent a specific Community Manager portal tenant instance. Lifecycle Manager is capable of automatically provisioning a single Service instances as multiple APIs across multiple portal tenants in a federated environment.
- **Legal Agreement**: a template designed to support publication of sharable legal agreements such as End User License Agreements (EULAs) enforced when APIs are exposed to business partners
- **API Access**: a template used to configure extended App to API access onboarding forms when used in conjunction with Community Manager.
- **SOA Reference Model**: a template used when publishing Lifecycle Manager Reference Models

In most cases, multiple asset constraint templates representing various stages of the software development lifecycle (SDLC) will be configured into Lifecycle Manager. Such templates establish incrementally increasing content mandates as an SDA advances through the SDLC.

2.6 Governing asset production and consumption

The Lifecycle Manager configurable process engine gives customers the ability to control the way their assets are produced, deployed and consumed within Lifecycle Manager. Customers can define their own asset management validations, processes and roles, which Lifecycle Manager then automates. For most deployments, this automation can occur entirely within Lifecycle Manager. Other customers may choose to use Lifecycle Manager automation to interact with their own external tools, workflows (both manual and automated) and other mechanisms. See Chapters 4 and 5 for more information about Lifecycle Manager capabilities to support creating and acquiring assets.

Lifecycle Manager is typically deployed in conjunction with Akana's runtime service and API management products including Policy Manager and Community Manager. In such deployments, Lifecycle Manager serves as a provisioning platform to automatically populate services and API definitions along with supporting constructs such as declarative provider-side policy assertions used to configure runtime enforcement platforms such as Akana's Network Director and DataPower, API productization content to be populated into Akana's Community Manager API portal, and even

deployment of executable images to appropriate runtime platforms via its integrations with devops and cloud provisioning platforms.

2.7 How can a user find Library assets?

Lifecycle Manager assets are owned by and made visible to users within a Library. One or more Libraries may be hosted on a single Lifecycle Manager installation, with each Library logically separated from other Libraries. Developers, business analysts, architects and other Lifecycle Manager users can search a Library to identify the business and technical requirements for application development and integration projects. Each Library may support different types of users. Each type of user may search the Library differently, using a mechanism based on the context of their work, what they know about the asset, or the Reference Model for the domain in focus. In addition, the search method can be selected based on the characteristics of the asset(s) being sought. Lifecycle Manager provides a number of search mechanisms:

- The **Asset Tree** organizes and presents assets based on their metadata into a dynamically generated multi-level folder view according to the user's preferences.
- **Queries** can be built using SDA metadata to quickly identify software assets of interest for a particular project. Queries can be stored and can incorporate many search criteria, including precise classifier-specific matching as well as text matching (including Boolean operators combining multiple text search strings) across any or all metadata elements of the asset. Advanced XPath-based mechanisms are built into Lifecycle Manager's query engine, giving customers a great deal of flexibility in defining precise query types without exposing the complexity of XPath or other query assertion languages to the end user. Query types are configurable, with a set of predefined types shipped as part of the product's default configuration. These types leverage the underlying XML and text-indexed search infrastructure and can present to the end user both single- and multi-field query forms.
- **Model-based searches** navigate through graphical Reference Models to select desired capabilities (e.g., functional aspects of the organization's business architecture) for direct search against Library content. Model-based searches can also be stored for future reference.
- **Search Alerts** allow users to mark any stored (persistent) search as "alertable," so that those users are notified of any new or re-published asset that meets its search parameters. This powerful alert mechanism helps users to automatically stay on top of new information (such as architectural patterns and best practices), improving communication within the development organization. Users can choose from email or RSS-style notifications for their search alerts.

See Chapter 5 for more information about Lifecycle Manager capabilities to support locating assets.

2.8 What is a Reference Model?

A Reference Model is a visual representation that illustrates how a particular system or business process (or a portion of the system or process) works.

Reference models can represent:

- Technical architectures
- Application architectures

- Business architectures and process models

Models can be built using different forms:

- UML (Universal Modeling Language)
- Microsoft Visio
- Business processes and use cases
- Free-format models rendered in HTML

These models can be imported into Lifecycle Manager to connect SDAs with elements of the model; as assets are captured or edited, their functions or features are mapped to the model. With this linkage, users can easily learn and understand which SDAs support or execute which model elements. Models are also useful as a form of architectural "heat map" that can be used by architects and business analysts to understand what aspects of the enterprise are well represented by reusable services and other shared capabilities and what aspects require additional investment.

2.9 Who uses Lifecycle Manager?

Many people within an organization use Lifecycle Manager: asset owners, asset consumers and people who provide Reference Models as a means to learn about and find assets. In addition, other people support Library activities such as project managers and other governance authorities who authorize asset publication and usage requests, usage controllers and Library administrators who configure the Library work environment and manage the Library itself are also active library users.

Lifecycle Manager provides tools to enable full and flexible access based on roles. The system automatically adjusts the user interface and functionality available for a particular user depending on their role. Role-based access is assigned at the user level and may apply Library-wide or may be scoped to the **Organizational Group** (Org Group) level to enable the user to complete tasks only for a particular group and its subordinate groups. See Chapter 4 for further discussion on this topic.

Standard Lifecycle Manager roles include:

- Asset User
- Asset Producer roles, including Asset Capture Engineer (ACE), Asset Publisher, and Asset Owner
- Usage Controller (the primary administrative role)
- Library Administrator
- Installation Administrator

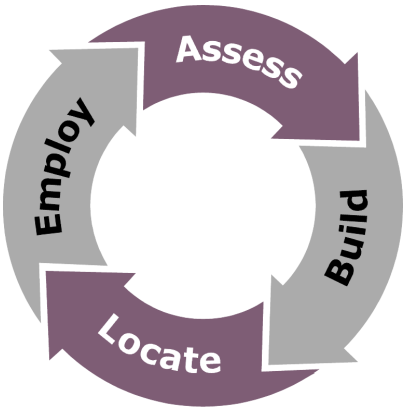
Lifecycle Manager administrators typically also introduce their own organization-specific roles into Lifecycle Manager for use in governance process automation. These roles can also be configured to control the ability to create and edit assets on a type-by-type basis and how asset details are presented in the browser-based thin client.

2.10 How can Lifecycle Manager fit my organizational and project team structure?

Lifecycle Manager is designed to work with all sizes of organizations – from a single team to a large organization, with multiple, geographically dispersed teams and cross-team assignments. Teams, organizational structures and shared work are an inherent part of Lifecycle Manager. In Lifecycle Manager, teams and organizational structures are called **Organizational Groups**. Shared work areas are known as **Projects.** Together, Organizational Groups and Projects can be used to designate organizational structures and define groups of users that work together on a common project. Organizations are also commonly used to establish shared governance scope over multiple teams (e.g., assigning an architect to each LOB represented in the Library). As project work groups are defined, group roles can be assigned to users, for example, indicating which users can create or change asset information available to the group. As discussed in section 2.9, user access to Lifecycle Manager can be defined according to a user's organizational group or their project. (Note: The tree-based inheritance structure of Organization Groups and Projects is useful when assignments of Profiles, Reference Models and other capabilities are common across more than one Project.).

The organizational support within Lifecycle Manager is very comprehensive. That said, you can start with a very simple organizational structure with Lifecycle Manager and grow into a deeper structure as needed. Akana professional services can also help determine how to best leverage this flexibility to suit a particular situation. Lifecycle Manager also enables distributed development through its federated library support, allowing large organizations to manage a consolidated set of SDAs through a physically distributed and interconnected set of Lifecycle Manager installations. See Chapter 4 for more information on use of organizational groups and federated libraries.

2.11 How do I get started?



Akana offers a proven methodology to help get started and realize a rapid return on a Lifecycle Manager investment. This approach is called the **enABLE** methodology:

Step 1: Assess

Understand what information the organization has -- what assets, what Reference Models, what asset artifacts, etc. Identify key initiatives and develop a roadmap to successfully leverage the valuable SDAs. This is discussed in Chapter 3.

Step 2: Build

Build the catalog of SDAs in the Lifecycle Manager Library and integrate the Library into the development environment and processes. This is discussed in Chapter 4.

Step 3: Locate

Roll the Library out to the asset users and educate them on how to effectively find the right SDAs for a given project. This is discussed in Chapter 5.

Step 4: Employ

Show the asset users how they can quickly understand and use the assets to deliver targeted projects. This is discussed in Chapter 6.

After completing the process, return to the Assess step to focus on assessing usage effectiveness of the Asset Library. Continue to define additional groups of assets to include in the Library and the next set of users for the Library.

An organization can conduct these steps on its own or work with Akana professional services.

3 Assessing a Situation

3.1 Setting the Strategy and Approach

Lifecycle Manager is typically implemented to improve enterprise visibility, use and traceability of key assets across an organization, be they APIs for external use, services in an SOA, reusable components, or knowledge assets such as architectures, design patterns and best practices. Lifecycle Manager can also be implemented to support a specific initiative that involves creating shared assets and rolling them out to intended asset users. Clearly understanding the initiative driving Lifecycle Manager's use is essential for planning an effective Library implementation strategy.

The Assessment

- The Right Pilot Project
- Getting Ready
  - What Assets?
  - What Models?
  - What Metadata?
  - What Users?
  - Development Process Changes
- Appropriate Staging

3.1.1 Assessing the Need

Regardless of how Lifecycle Manager is to be used, basic upfront assessment and planning is critical. Before the Lifecycle Manager Library is deployed:

- Document existing development processes so that the Lifecycle Manager asset search, asset capture and asset governance functions can be aligned with these processes.
- List candidate SDAs.
- Define the information required for each asset type.
- For each asset type, note the source of the information associated with it, what roles need to review it and when it needs to be reviewed.
- Determine what models describe the SDAs. (This may include using existing models or creating new models as appropriate).
- Designate the organizations or projects that will use the Library.

3.1.2 Determining the Roll Out

After completing these steps, initiate a staged roll out plan. For each stage, address the following:

- Business objectives and value/business owner.
- Specific asset and model identification (including ownership and contacts).
- Asset users/organizations and project requirements.
- Asset configuration (including where the Library will be hosted and how it will be administered).
  - Organizational and project structure set up
  - User role definitions
  - User id creation (typically including LDAP server / SSO integration)
- Governance process configuration.
- Asset Library integration with various asset sources (where asset information will come from).
- Development process and tool integration (including integration into chosen IDEs, intra-enterprise, or inter-enterprise development portals).

- Training plans and roll out (Akana Professional Services can provide training for end users or trainers).
- Communications launch.
- On-going communications and management relative to business objectives.

3.2 Ideal Pilot Projects

Projects with defined assets that have been built for reuse (e.g., API and SOA initiatives, component-based development teams) are prime candidates for initial Library projects. If assets have been architected to be used by others (executable or knowledge assets), they are likely to have models and other documentation that is ready to use, clear asset owners, and defined asset users.

Over time, other assets and user groups can be added to the Library. Lifecycle Manager supports an **asset governance** approach that allows asset owners to submit assets for inclusion in the Library. Submitted assets can then be directed to other roles responsible for reviewing and publishing the Library content, or simply directly provisioned to the Library and associated runtime platforms as part of a devops automation initiative. The submission mechanism also serves as a way to incorporate user feedback into the process of defining and creating new assets.

As part of Lifecycle Manager delivery, Akana will provide you with a series of SDA reuse best practices that include both organizational and asset production and consumption process guidelines. These best practices include asset reuse ROI calculations based on work by Dr. Jeffery Poulin, a leading industry expert on SDA reuse.

3.3 Setting Realistic Goals

The “start small, grow fast” approach is very appropriate for Library roll out. Starting small enables understanding of what the Library can do and how it will work best in an organization on a broader scale. Before finalizing an enterprise-wide implementation plan, focus on gaining experience in several areas, including:

- Defining the Asset metadata (schema) needed for each asset type.
- Configuring and integrating Asset sources (i.e., where the asset information will come from).
- Selecting and using Reference Models.
- Handling asset requests (i.e., asset consumption) and asset submissions (i.e., asset production).
- Reporting metrics and ongoing assessments.
- Establishing and refining development processes, including approvals, asset acquisition, reviews, etc., and defining and configuring associated asset governance automation processes for asset submission and asset acquisition

Much will be learned by using the Library in an organization and actively listening to user needs. Akana professional services can also help with start-up, pilot projects, results assessments, and tuning for follow-on implementation.

4 Building the Library: Integrating Into an Existing Development Environment

Building the Library to fit the needs of an organization is the most important step in deploying Lifecycle Manager. Lifecycle Manager was designed with the flexibility to integrate with any development process regardless of whether it is RUP (Rational Unified Process), Agile, or any other process in use within the software community today.

Key integration points for the Library are:

- Capturing reference and architectural models.
- Capturing/updating asset information.
- Finding assets for use during development.

Building a Library and integrating it into a development process and environment is likely to be an on-going and staged effort. First, determine what information will help the development teams understand and use the assets. Also, determine how the users want to search the Library. Finally, plan the most efficient way to load assets and models, and keep them current.

Building a Library

- Development Tool Integration
- Asset Template Definition
- Reference Model Selection
- Asset Production Processes
  - Linking to Asset Sources
  - Asset production governance
  - Provisioning Operational platforms
- Defining Users, Organizations and Projects

The following sections look at some of the major concepts that must be understood to define how the Library will fit into an environment.

Note that even though Lifecycle Manager is highly flexible, it ships with many default settings to speed installation and initial operation. The end of this chapter summarizes the quickest way to get started.

4.1 Development Tool Integration

Akana provides various integration points to popular development tools. This integration helps users find assets, capture and edit them, and import Reference Models. In the future, Akana will integrate Lifecycle Manager with other popular IDEs and related development tools based on industry demand.

Integration Type	Product	Description
FINDING ASSETS: Launch Lifecycle Manager for Asset Searching	Microsoft Visual Studio, IBM Rational Application Developer, Eclipse, other Eclipse-based commercial IDEs	Add-in / Plug-in
IMPORTING REFERENCE MODELS: Reference Model Creation	Microsoft Visio	Plug-in
CAPTURING/EDITING ASSETS: Project/Solution/Asset Capture and Mapping to Reference Models	Microsoft Visual Studio, IBM Rational Application Developer, Eclipse, other Eclipse-based commercial IDEs	Add-in / Plug-in
PROMOTING ASSETS: Devops Integration	Jenkins, TeamCity	Governance automation tasks

4.2 What Data Is In the Library for a Given Asset Type? Using Asset Templates

Templates are used in the Library and the capture wizard within the Lifecycle Manager IDE Plug-in for validation during asset creation, updating, and publication. Templates are created in the form of XML documents, and must meet the requirements of underlying asset and template schemas to which they apply. Once created, templates are made available to the Library by uploading them via Lifecycle Manager's Configuration Designer. Configuration Designer is discussed further in a later section of this chapter.

4.2.1 Template Types

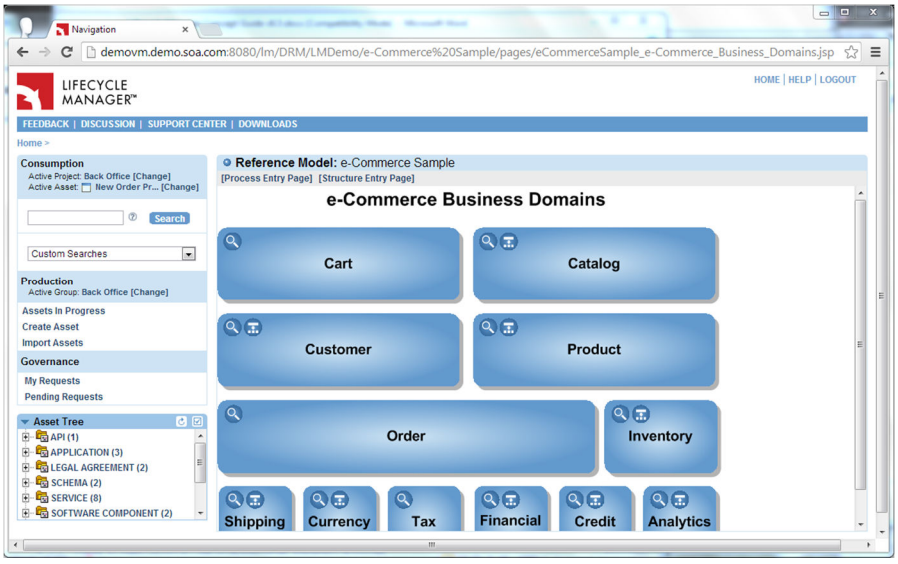
Lifecycle Manager specifies two types of asset templates. Each template type has a specific use.

1) **Global Definition Template**. The Library has a Global Definition Template used to define the superset of asset metadata elements (i.e. classifiers, asset relationships, and artifacts) available for assets published to the Library. The Global Definition Template serves as the basis for all other templates.

2) **Constraint Templates.** Constraint templates define the specific elements used for each type of Asset. The Library can have any number of Constraint Templates that extend the Global Definition Template, but it must have at least one Constraint Template. Once these are uploaded to the Library, they are used to validate Assets during the capture, edit, and publishing process. In addition to specifying the metadata elements used for an Asset Type, a Constraint Template can designate additional constraints on cardinality, value lists, and default or override value settings during asset creation and subsequent editing. Constraint Templates can also introduce conditional content rules based on the value of specific classifiers in an asset instance. If defined correctly, Constraint Templates can greatly ease the process of asset capture by ensuring the right asset information is gathered, and by predefining certain pieces of information that are likely to be well-known or constant for a particular type of asset. Constraint templates can also enforce incremental information gathering across an asset's SDLC (e.g., requiring that a test plan be defined at design-complete phase and that test results be provided before exiting the development-complete phase). Constraint templates can be configured to put control over template change fully under automation (i.e., governance processes automatically switch templates for an asset only when the asset exits a governance phase), under user control with guidance (i.e., each template specifies the allowed downstream templates that can be selected for asset transition) or fully under user control.

4.3 Reference Model Definition

The objective of a Reference Model is to aid in the understanding of assets and how they work together. Reference Models can include application architectures, technical architectures, business process definitions, and other types of models that provide a representation of the domain.



While models can link to assets in many ways, one common approach is to create a "visual domain taxonomy" through graphical images that can be navigated to initiate Library searches. The traditional layered architecture block diagram is well suited to this approach. One advantage to this taxonomic approach to Reference Model development is that assets can be automatically mapped to model elements simply by populating the necessary taxonomy information into the asset metadata. As discussed below, Lifecycle Manager provides integrated tooling combining Microsoft Visio and Eclipse to aid its customers in quickly and efficiently developing such models. Other tools and techniques can also be used to develop and deploy Reference Models into Lifecycle Manager.

4.3.1 Tools to Help Build a Lifecycle Manager Reference Model

The HTML used in a model can come from many sources. Two popular sources are Visio diagrams of business process or architecture models, or UML models created with modeling tools. Visio-based model definition is natively supported by Akana tooling (discussed below). In any event, the HTML pages for the Lifecycle Manager model must be created with the links between the pages defined. Any HTML-editing tool can be used to create model pages to import into Lifecycle Manager.

When a Reference Model is defined, it will consist of a set of HTML pages and an associated XML file that defines the structure of the model. This pair of artifacts is then published into the Library. Once published, the model can be accessed to allow asset capture engineers to map assets being captured or updated to the models. Ultimately, the model can be accessed when doing a model-based search to allow the asset user to specify the requirements they are working to fulfill. Note that Lifecycle Manager has tools to help with the creation of the model artifacts and publication into the Library. In particular, Lifecycle Manager provides an Eclipse-based reference model development plug-in integrated with Visio.

4.3.2 Two Reference Model Approaches

There are two approaches to consider when defining a Reference Model: loosely coupled or tightly coupled. Coupling refers to how well Library assets are likely to match the model. Both approaches are described further below with examples of when to use them.

4.3.2.1 Tightly Coupled Models and Assets

A tightly coupled model is typically a model that was used to define the assets in the Library. It is "bottom-up" in the sense that it was built based on assets already in the Library, or created to define specific assets needed for the Library. As such, assets in the Library will have a very high affinity to the model, almost always yielding a 100% match between functional requirements and asset functionality.

The benefit of a tightly coupled model is that it provides a valuable navigation and search mechanism for users. They can peruse the model to understand the domain and then quickly search to see the assets that fulfill the architecture. Such models can also be used as "progress indicators" against a strategic architectural initiative: the greater the progress, the more portions of the model will result in successful asset searches.

4.3.2.2 Loosely Coupled Models and Assets

In contrast with a tightly coupled model, a loosely coupled model is created from the "top-down" -- independent of assets in the Library – to define goals or roadmaps that may have not yet been implemented. Consequently, Library assets will have varying affinity to a loosely coupled model. Examples of this approach would include models that define industry-standard processes and assets, such as third-party components, that provide function for the same domain, but may or may not conform to the standards depicted in the model.

4.4 Creating and Working With Assets

There are two important steps for adding asset information to the Library. The first step is to **capture** the asset metadata and the second is to **publish** it into the Library.

4.4.1 Asset Capture

Capturing assets is the process of entering and creating references to the information that defines an asset. Once this process is complete, the asset is released for publication into the Library.

Assets can be created or updated in four ways, via the:

- Lifecycle Manager IBM Rational Application Developer or Eclipse Plug-in / Microsoft Visual Studio Add-in Capture Wizard and Asset Editor.
- Lifecycle Manager Library (browser-based) user interface (including inline asset creation during asset edit via the Add Relationship action, Import Center support for XPDL, BPEL, WSDL, XSD and XMI-based assets, configurable search results processing actions resulting in asset creation).
- Lifecycle Manager AnySource™ Asset Adapter Toolkit to link to asset sources and drive programmatic collection of asset information.
- Tool-based or file-driven bulk creation/updating techniques.

Each approach uses the Asset Template to define and validate the information captured. See section 4.2 for more explanation of Asset Templates. Each approach also generates an XML document which holds the asset metadata and resides in the Library's asset catalog This document is automatically indexed and made available to the Library's XPath-based query engine, and can also be directly retrieved and viewed from the thin-client UI by users with the Usage Controller role). When an asset is published, the Library reads the corresponding metadata and uses it to create the asset information in the Library.

The **Asset Capture Engineer** is the Lifecycle Manager role which is authorized to capture assets. For each asset, the Asset Capture Engineer will capture the following information (asset metadata):

- **Overview Information**: name, version, overview, description, aliases
- **Classifiers**: asset type, supported platforms, functional domain & sub- domain, support level, … (note that both simple and compound (i.e., multi-level) classifiers are supported by Lifecycle Manager)
- **Artifacts**: asset models, source code, binaries, user documentation, test scripts, models, …
- **Asset Relationships**: previous versions, prerequisites, co-requisites, suggested usage, …

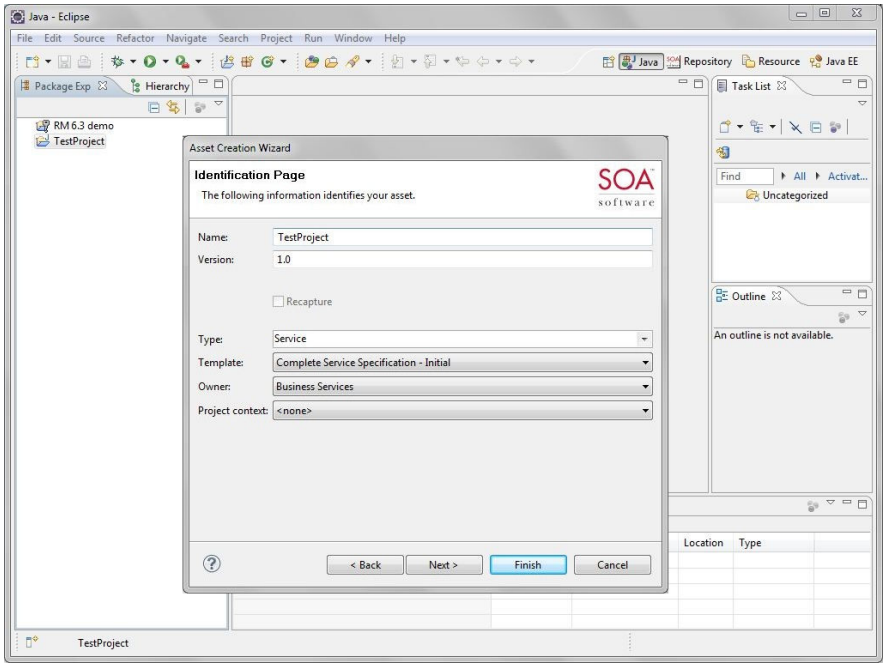
Assets are always created and edited within the context of an owning organizational group. An Asset Capture Engineer selects the asset's owning group from the list of groups for which he or she has been granted the Asset Capture Engineer role. Asset Capture Engineers can easily determine the group currently in force by viewing the Active Group field within Production section of the left-nav bar of the thin-client UI or by viewing the equivalent field under the Production node within the plug-in's Library Explorer view.

Other roles can be combined with the Asset Capture Engineer role to establish role-specific asset editing capabilities, including both the ability to create and edit assets of any particular type and constraints over the asset metadata available for editing. For example, an organization may desire that a Business Analyst be allowed to edit only business-related metadata for Business Process type assets only, whereas an Architect may be allowed to edit all metadata defined for any type of asset. Metadata elements can also be configured as read-only for any or all roles (e.g., a classifier value that is automatically calculated by Lifecycle Manager automation and should not be changed by any user, technical metadata that should not be edited by a user with the Business Analyst role).

Unpublished (pending) changes to an asset can be easily reverted to the last published values as needed in the event of an error in updating the asset, a change in development plans, or other reasons.

4.4.1.1 Capture Wizard and Asset Editor

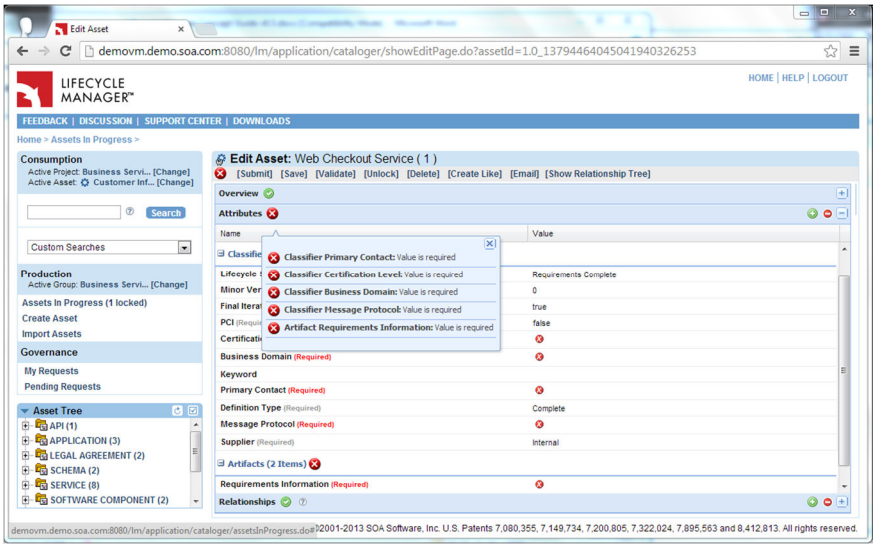
The Lifecycle Manager IBM Rational Application Developer and Eclipse plug-in Capture Wizard and Asset Editor is an integrated facility that supports rapid capture of Projects as Lifecycle Manager assets, as well as easy asset editing. Likewise, the Lifecycle Manager Visual Studio Add-in provides similar support for Visual Studio projects and solutions implemented within that Integrated Development Environments (IDEs). These Add-in / Plug-ins use asset templates to access the asset catalog, and provide two capture modes:



1) **Capture Wizard.** The Capture Wizard allows IDE users to quickly capture a Project or Visual Studio Solution as a Lifecycle Manager asset. The wizard extracts asset metadata from the selected project or solution and (for Visual Studio) associated assembly metadata, combining that metadata with user-specific properties managed by the Lifecycle Manager Add-in / Plug-in. Asset capture using the wizard can be as simple as a "one-click" process, or the user can fine-tune the automatically gathered metadata with additional asset information through a series of wizard panels. Upon completion of the wizard, the asset can be submitted to the Library for publication or can be immediately transferred to the Plug-in / Add-in Asset Editor for further refinement. Open-ended (i.e., non Project- or Solution-based) assets can also be captured through the Capture Wizard.

2) **Asset Editor.** The Asset Editor allows IDE users to edit any Lifecycle Manager-managed asset. Advanced features of the Asset Editor include drag-and-drop support for adding asset artifacts and establishing asset-to-asset relationships. Users can also edit other aspects of asset metadata using standard IDE user interactions including drop-down lists, editable text boxes, dialogs, and wizards.

4.4.1.2 Library Interface for Asset Creation and Editing



Assets can be created or edited using the Library interface. This approach is most useful when creating or editing individual assets (or a small set of assets).

The Library interface also supports direct importation of XPDL, BPEL, WSDL, XSD and XMI files for purposes of creating Business Process, Service and Schema assets (as well as assets derived from UML model parsing rules) via the Import Center feature. This feature is described further in Chapter 8 of this guide. Users may also be able to select assets from the Search Results page for processing which results in asset creation (e.g., grouping a collection of selected executable assets into a release by creating a Release asset which references the selected assets).

Retrieval of metadata values from external sources

Organizations often desire to source certain metadata values from external systems. Lifecycle Manager supports such sourcing via its extensible Value Source framework. Simple Value Source implementations provided by Lifecycle Manager include OWL Lite-based taxonomies, WSDL elements (such as declared operations) and previously entered values (often referred to as “folksonomies”). In addition, Lifecycle Manager provides prebuilt interactive Value Sources for LDAP (commonly used to select contact information) and SQL (suitable for any externally-sourced enumeration). Users interact with these Value Sources via imbedded and configurable wizards by entering query criteria and in turn are presented with a list of valid values conforming to that criteria returned from the Value Source. Custom Value Sources can also be implemented and deployed into the Value Source framework.

Viewing pending changes to an asset

Asset editors using the thin-client Library interface may compare an asset being edited against the currently published values for that asset. Selection of this action presents asset additions, deletions and modifications in digest form. Asset editors can also compare modified WSDL and other XML document-type artifacts within such assets using Lifecycle Manager’s built-in artifact comparators. Customers can implement and deploy their own custom comparators within Lifecycle Manager for these and other artifact types as desired.

4.4.1.3 AnySource™ Asset Adapter Toolkit

Using the AnySource™ Asset Adapter Toolkit (AnySource) to create assets is most useful when capturing *large quantities* of assets and when automating asset updates based on changes to asset information and artifacts held by other tools outside Lifecycle Manager. These external tools/repositories are referred to as **Asset Sources**, and may include source code control systems, document control systems, defect tracking systems, etc.

AnySource provides an XML-driven framework based on Apache Ant that is used to define and automate the asset capture process. In addition, AnySource capabilities can be directly accessed through programmatic (Java or .NET) and command line means if desired. Underlying these capabilities is a series of SOAP-based Web services that connect Lifecycle Manager's Automation Extensions client to the Lifecycle Manager Asset Library.

The AnySource framework uses AnySource parsers to extract and interpret information from the various files that make up the asset. Parsers are typically used to analyze and gather asset metadata for eventual inclusion into the asset being assembled, and are invoked by XML-driven tasks that define the asset capture process. AnySource then directly publishes the asset information to the Asset Library where it can be automatically made available to asset users or, optionally, it can be reviewed, corrected and augmented by ACEs as appropriate prior to submission for role-based governance approval. Akana provides out-of-the-box parsers for open-ended text-based and XML-based processing, as well as specialized Java sources and WSDL parsers. Custom AnySource parsers can also be written in Java, optionally taking advantage of Java's ability to call out to other languages of choice, including scripting languages such as PERL, Python, etc.

In addition to the core AnySource toolkit, Akana provides prebuilt AnySource adapters for leading Source Change Management (SCM) systems such as GitHub, CVS, Subversion, Serena Professional (PVCS) and Dimensions, Rational ClearCase (including ClearCase/ClearQuest deployed in UCM mode), Rational Team Concert (RTC), Perforce, and Microsoft Visual Source Safe and Team Foundation Server (TFS) version control. A general purpose WebDAV-based adapter is also available. These adapters allow assets and artifacts to be loaded into and retrieved from the Library. Additional prebuilt adapters will be provided in future releases of Lifecycle Manager based on industry demand.

Documentation provided with AnySource includes:

- A User Guide which describes how to install and use the AnySource Toolkit
- A Developer Guide which describes how to write AnySource extensions
- Best Practices documents that provide Source Change Management (SCM)-specific guidance for usage and deployment of the Toolkit

AnySource Artifact Adapter

Because many organizations have widely varying asset sources (e.g., multiple SCM systems, defect tracking systems, etc.) and because often these systems of record are not integrated from an authentication perspective and/or do not support modern web-based access methods, Lifecycle

Manager provides the AnySource Artifact Adapter. Even when integrated authentication is in place, organizations often establish project-specific access rules in keeping with SCM best practices, thereby preventing potential asset consumers from directly accessing SCM-hosted files. When configured properly, the AnySource Artifact Adapter provides seamless access to artifacts throughout the enterprise, eliminating the need for the Lifecycle Manager user to sign on to each system separately and serving as an SCM proxy for asset consumers. Lifecycle Manager manages the authentication credentials required to connect to and retrieve artifacts from the various systems of record integrated with a Lifecycle Manager installation, either through out-of-the-box adapters provided as part of the Lifecycle Manager installation or custom adapters that can be written to support essentially any form of access APIs provided by those systems. Prebuilt adapters include ClearCase, TFS, RTC and HTTP basic auth. In conjunction with Lifecycle Manager's IDE plugins, ClearCase, RTC and TFS adapters support importation of SCM controlled file references into SCM controlled projects, thereby ensuring that the project has access to the most current source or executable content. Lifecycle Manager also provides a SOAP-based adapter framework for use by customers to encapsulate other systems of record.

4.4.1.4 File and API-based Asset Creation and Update

Lifecycle Manager provides numerous file-based importers that can be used to create and update assets. These importers are implemented as extensions to a general purpose importer framework that can be extended as needed by specific customers. Out-of-the-box importers include:

- Service/API descriptor documents in WSDL, Swagger, RAML or WADL format
- Schemas in XSD format
- Business process definitions in BPEL or XPDL format
- UML models in XMI format
- Multi-asset import via tab-delimited file format

Of the above list, Lifecycle Manager also provides REST API-based support for tab-delimited files to enable automated scripting of bulk content creation or update. Lifecycle Manager also provides a special-purpose REST API that enables automated creation of an API described in any of the above listed descriptor document formats.

4.4.2 Artifact Management

Asset artifacts are often simply managed by Lifecycle Manager as independent work products resulting from asset development. In other words, such artifacts are accessible through the Library but have no effect on Library behavior. However, some artifacts may require or be able to take advantage of special handling by Lifecycle Manager, perhaps because of the sensitive nature of the asset, because their content and format enables additional capabilities within Lifecycle Manager and the tools with which Lifecycle Manager integrates, or simply because of their importance to the user. Lifecycle Manager supports the following specialized artifact-handling capabilities:

- **Service/API Definition Language Artifacts**
  In addition to traditional WSDL documents used to specify SOAP-based services, Lifecycle Manager provides native support for modern API definition languages such as Swagger, RAML

and WADL. Services and APIs specified via such artifacts can be validated against organizational compliance policies such as naming conventions, proper use of imported/included schemas, etc. and can also be automatically provisioned to Akana's Policy Manager runtime registry and Community Manager API portal.

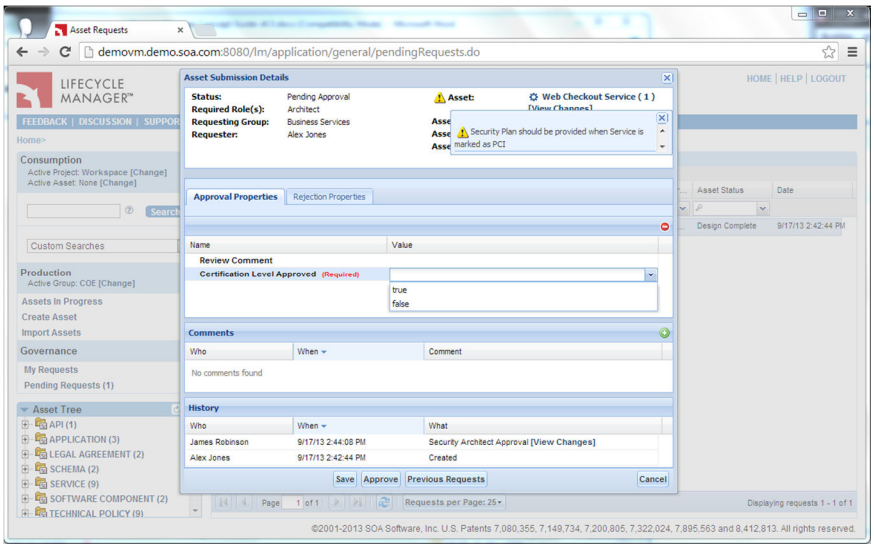
- **Queryable Artifacts**
  A queryable artifact is one whose content is included in the Lifecycle Manager search index. For example, consider a requirements document filled with domain terminology. By specifying artifacts of type "requirements" as queryable in the Lifecycle Manager Global Definition Template, Lifecycle Manager will automatically index the contents of all artifacts of that type at the time of asset publication. Users can then selectively include these indexed artifacts in their adhoc searches and asset queries (see Section 5 for additional details). XML-based artifacts are particularly well-suited for Lifecycle Manager's queryable artifact capabilities, as their content is automatically indexed to support both standard text-based queries as well as XPath assertions configured into Lifecycle Manager's query infrastructure.

- **Private Artifacts**
  Asset templates can specify one or more asset artifact types as "private". While the presence of private artifacts is made known to asset users, such artifacts cannot be retrieved by a user until their associated asset has been acquired and fully registered by the user's project. This feature can be used, for example, to allow organizations to prevent access to sensitive artifacts until the appropriate reviews have occurred. Other organizations may wish to restrict access to run-time components until the approval process has been completed, while still granting access to asset documentation.

- **Unpackable Artifacts**
  Browser-based HTML content such as Javadoc is often delivered in complex forms, with imbedded graphical elements and multiple linked pages assembled into a composite module. Lifecycle Manager allows users to store such complex HTML content within Lifecycle Manager in zipped form, with Lifecycle Manager handling all the presentation details to the user.

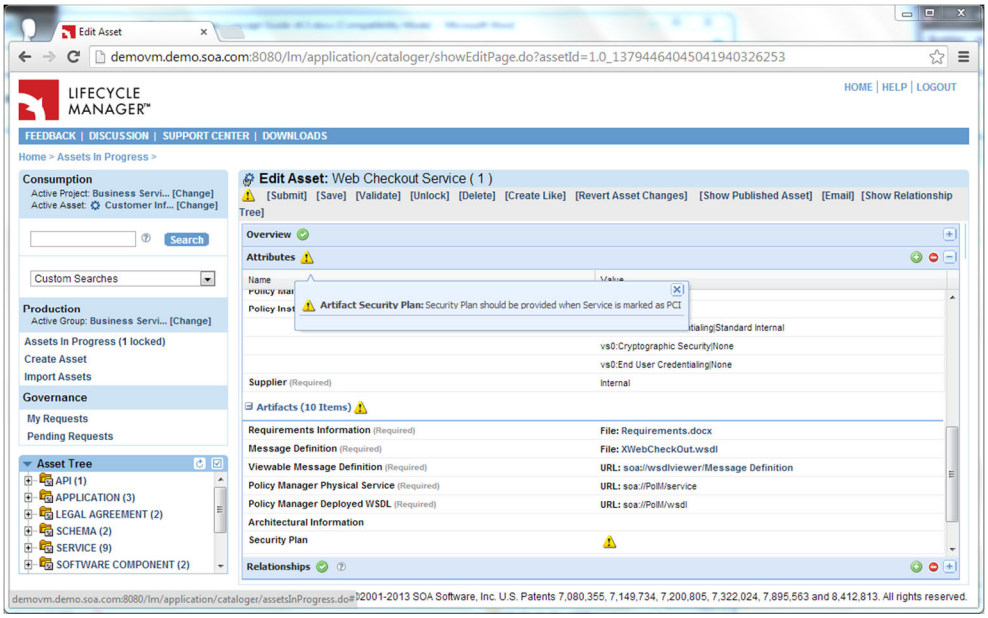
- **Email-Based Dynamic Artifacts**
  Informal design reviews and other development discussions often occur via extended email threads spanning multiple team members. Such email threads can be initiated from an asset via use of Lifecycle Manager's email-based artifact type. As the email is sent and responded to by the community of interest, Lifecycle Manager automatically updates the asset with the latest email in the discussion chain. The resulting artifact serves as a historical record of informal discussion and decisions that are otherwise commonly lost to development teams.

4.4.3 Validating, Reviewing, and Publishing an Asset



As was mentioned above, all of the capture approaches yield an XML document which holds the asset metadata and which resides in the asset catalog. When an asset is published, the Library takes the information in the catalog and uses it to create asset Library entries. These entries are held in both XML and relational form in the Library's database for rapid search and retrieval. However, before the asset can be published into the Lifecycle Manager Library, it must complete the asset submission governance process appropriate for its contents and organizational ownership. At one extreme, the submission governance process can be instantaneous (i.e., minimal content is required by Constraint Templates, and when an asset is submitted into the Library, it is automatically passed along to the Library's asset publication mechanisms). At the other extreme, an organization might use Lifecycle Manager's Constraint Templates and governance automation processes to configure a combination of incremental content mandates aligned with a multi-stage review and signoff process for an asset, with this process consisting of a mix of automated validation actions and manual review steps.

4.4.3.1 Edit-time Validation



As discussed in section 4.2 above, Constraint Templates are used within Lifecycle Manager to establish basic content validation rules for assets, including:

- Cardinality (mandatory/optional, single or multi-valued)
- Valid value lists
- Default values
- Override values (as an asset progresses through its SDLC phases)

In many cases, such validation rules are sufficient to guide Asset Capture Engineers in correctly and completely populating assets. However, other situations may require more complex validation rules. To support these situations, Lifecycle Manager provides a synchronous (edit-time) validation framework. This framework supports both declarative and algorithmic validation rules and can raise detected conditions to the user in the form of severe, warning or informational messages. Declarative rules are expressed via a specialized XML document, while algorithmic validation rules have access to the full power of Lifecycle Manager's local Java-based APIs via Beanshell or Jython scripts or native Java extension classes. Examples of declarative validation rules that can be easily configured into Lifecycle Manager include:

- Content format validation through Perl-style regex expressions (e.g., asset version number must be of the format V<positive integer>.<non-negative integer>)
- Conditional content enforcement (e.g., if a Service asset is designated as SOAP, a WSDL is mandated; if designated as REST, the REST operation dialog is presented and mandated)
- Conditional relationship validation (e.g., if asset-type is set to "Service" then consumes-schema relationship must be established and linked to a valid asset of type Schema)
- Cross-classifier validation (e.g., if classifier service-source is set to "packaged application" then classifier provider must be set to a valid packaged application ISV)

Algorithmic validation rules often retrieve other assets and/or information from external sources to determine asset validity. They may also execute calculations to derive and populate asset metadata values. Examples of prebuilt algorithmic validation rules provided as part of Lifecycle Manager include:

- Parsing and validating of imports and includes specified by WSDL and XSD artifacts to generate acquisition-type consumption relationships between Service and Schema assets (including support for enforcing use of governed schemas only)
- Service and schema namespace uniqueness validation
- Service port type and binding uniqueness validation
- Automated establishment of asset relationships based on classifier values (e.g., if a service is of type Data Service automatically associate a specific Technical Policy Set asset with the service to drive automated operational policy assignment as discussed in section 4.4.3.5 below)

Validation rules can be configured to run during asset creation, asset new version or copy creation, asset update (i.e., at time of asset submission or via invocation of an explicit Validate action by the end user), application of a new template to the asset (e.g., when automated template change occurs as the result of governance process automation) or asset deletion.

4.4.3.2 Governance Process Automation

Lifecycle Manager's governance process automation capabilities enable administrators to:

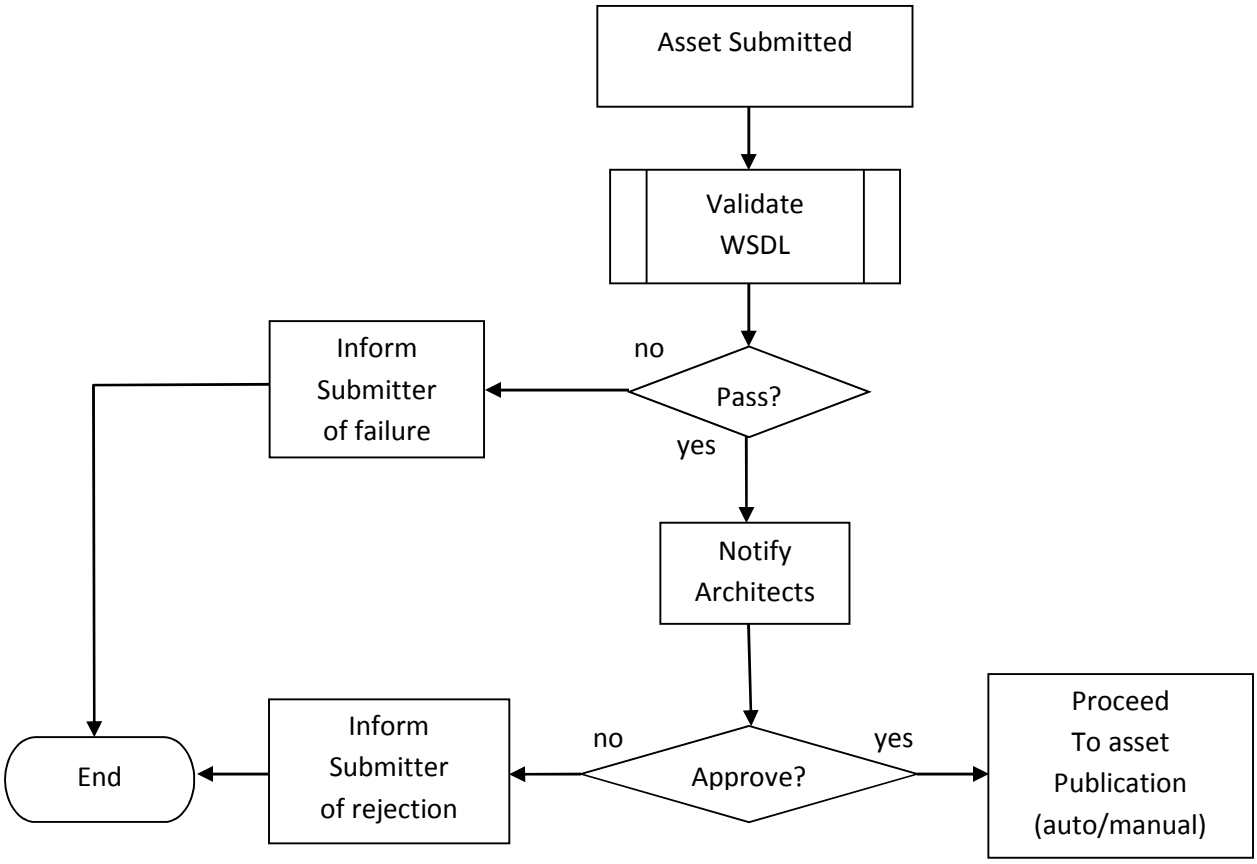
- Define custom roles
- Assign roles to users within organizational structure
- Define custom asset production governance/review processes, including:
    - Automated asset validation (including asset metadata and properties), parsing and post-processing activities using both prebuilt and custom scripted automation tasks

  - Role-based and user-specific review flows, including both comment-only and approval request types and notification and signoff of asset changes by existing consumers of that asset if desired
  - Checklist-style request fields made available for review and audit activities, including advanced capabilities such as:
    - Nested field support with subfields controlled by parent field values (e.g., if an asset is approved conditionally, a customer may wish to gather additional information on the conditional restriction being established)
    - Visibility and cardinality control based on request approval or rejection (e.g., require a reason code if a request is rejected, require an approval level if a request is approved)
  - Automatic or manual final asset publication with:
    - Configurable automatic discussion forum topic creation, including optional configuration of an inline "comments" section exposed directly on the thin-client Asset Details page
    - Automated contact list population based on org structure
    - Designation of selected artifacts as "private" (not retrievable/downloadable until project approval granted)

For example, consider the following asset production governance scenario: an enterprise has defined their Web services production SDLC to include reviews at these points in the lifecycle:

- ***Requirements complete:*** all business requirements documented and initial service definition specified, allowing **business analyst** and **architecture** reviewers to validate service against business context
- ***Design complete:*** Interface WSDL provided, implementation approach defined with sufficient documentation (e.g., UML design models completed, relevant legacy APIs identified, test plan defined) to allow **architecture, security** and **test lead** reviewers to validate design against technical and application/integration contexts
- ***Implementation complete:*** Service implemented and deployed in a test environment, with sufficient supporting documentation (e.g., sample client code, automated test cases, usage guide) to enable a potential consumer to understand the service; **architecture**, **performance** and **operations** reviewers complete final review before deployment

Governance roles and processes can be configured to automate each of these review points. A simple design complete governance process might look something like this:



Each of the other review points would have their own governance roles and processes defined, with filters configured to route a submitted SDA to the appropriate process based on asset metadata and organizational structure.

Users involved in a governance review process are automatically notified of pending requests via an email which contains an imbedded URL to the request details. Reviewers can also access via the Lifecycle Manager thin client a list of pending requests as well as requests that they have acted upon in the past. Upon opening of a pending request, a user can easily view prior governance requests and differences between governance requests, as well as change history for the asset currently under governance. They can choose to approve or reject the request at that time or can simply edit the request and save for later approval or to pass along to another governance user for review and final action. Depending upon the process configuration, a flagged approval could even trigger a dispensation reminder at some point in the future (e.g., when the business sponsor requires a non-conformant API to be published to meet an urgent business need, an architect will likely want to initiate a post-production review of the API to determine the best way to bring it into compliance). Users with Usage Controller

administrative authority are given access to an additional page which lists all requests in the library. All of these pages provide appropriate sorting and filtering options to allow the user to quickly find the desired request based on submitter, status, request type, and other parameters.

Asset submitters can view their active and inactive requests through a similar set of pages in the Lifecycle Manager thin client. Active requests show submitters the currently pending governance approval role along with Lifecycle Manager users having that role-based authority to approve or reject the submission request, thereby making it easy to ping approvers for requests that are not being expeditiously processed. Submitters needing to resubmit an asset (e.g., because a role-based reviewer rejected the original submission) can choose to prime their resubmission with request field values from the immediately prior submission request for that asset.

4.4.3.3 Publication Templates

Once the submitted asset passes through the configured governance review process (if any), it proceeds to asset publication. Just like the capture process, templates are used for asset publication to help define and expedite the process. The publication template can come from the group or person(s) that owns the asset(s) to be published. Group-derived templates can be further selected through metadata filtering rules (e.g., assign a publication template that restricts executable jar file access until consumption approval is granted for assets designated as enterprise certified, while allowing users to directly access such files for noncertified assets). While publication is typically invoked automatically, it can also be manually based on the library's configured publication template settings. In the case of manual publication, assets ready for publication are queued up as **publication requests** for the **Asset Publisher**. For each asset, the publisher provides some additional information when publishing an asset. For automated publication, this additional detail is included in the publication template.

4.4.3.4 Slack Integration

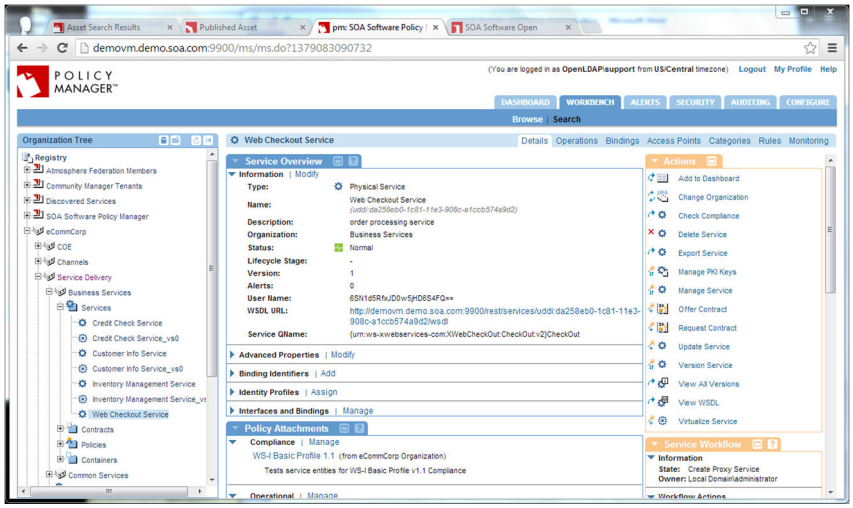
Slack is a popular team-based communications tool increasingly used by developer communities. As part of its process automation capabilities, Lifecycle Manager supports the following configurable Slack-based notifications:

- Notification of role-based approvers via Slack channel – for example, a customer can configure an "architects" channel to receive signoff requests for service provisioning or an "apiproductmanagers" channel to receive API provisioning requests.
- Notification to users when their submission request is approved or rejected.
- Notification to users when their submitted asset is published.
- Notification of publication events via Slack channel – for example, a customer can configure a "services" channel to receive postings whenever a Service asset is published

Each Slack notification imbeds a url to the relevant Lifecycle Manager page or dialog, thereby enabling Slack users to quickly access dev-time Service and API information of interest.

4.4.3.5 New Version and "Create Like" Processing

An important aspect of an ongoing asset management initiative is new version processing. Lifecycle Manager provides customers with a great deal of flexibility when creating new asset versions. By default, Lifecycle Manager automatically makes a deep copy of the asset being versioned (including copies of any by-value artifacts contained by the asset). When more control over new version content is required, customers can designate a specialized "new version" constraint template that can be configured to default, override and clear the asset's metadata fields and associated artifacts as required. In addition, specialized synchronous validation rules can be automatically invoked at time of new version creation (e.g., to automatically format the version number of the newly created asset according to the organization's version numbering rules). Lifecycle Manager also offers users a "Create Like" action which has behavior and configurability similar to new version creation. The primary difference between "Create New Version" and "Create Like" actions is that "Create New Version" automatically establishes a predecessor/successor relationship between the existing and new asset (which is used to generate notifications to subscribers and consumers of the existing asset) whereas "Create Like" does not establish such a link.



4.4.3.6 Synchronizing Services with Policy Manager

In an integrated configuration, Lifecycle Manager automatically synchronizes service definitions (i.e., WSDLs) and selected metadata with Akana's Policy Manager product. Lifecycle Manager can also be configured to automatically advance the operational governance state of the service in Policy Manager, thereby signaling to operational administrators when the handoff from development to operations control for a service has occurred. As a service progresses through operational staging to production, these transitions can in turn be synchronized back to Lifecycle Manager in the form of service state updates.

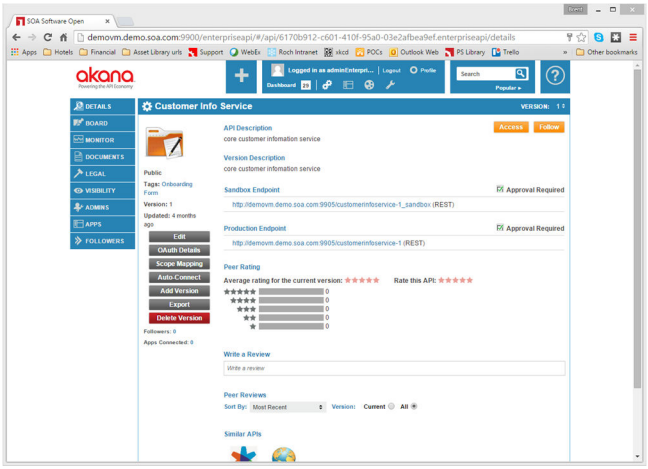
Lifecycle Manager is capable of validating, governing and synchronizing WSDLs developed through top-down modeling practices (i.e., WSDLs as independent work products referencing independently governed schemas) and through bottom-up tooling practices (i.e., WSDLs bundled with implementation schemas generated through development tooling and referenced via relative path). In the latter case, such services are typically packaged as zip files for association with services under governance; Lifecycle Manager automatically extracts the WSDL and associated schema documents from such files and builds the XML dependency graph prior to synchronization with Policy Manager. Customers using Akana's Network Director or Policy Manager for DataPower components in concert with Policy Manager to

create proxy virtual services for consumption purposes can choose to synchronize metadata representing operational and QoS policies for each virtual service from Lifecycle Manager to Policy Manager. This metadata in turn is used by Policy Manager to automatically assign designated audit and security policies to each proxy virtual service as it is generated in Network Director. For instance, an organization may define three audit levels (e.g., normal, high and diagnostic) and three modes of consumer authentication (e.g., standard internal consumer, rich-client internal consumer, and B2B) as part of its operational service governance framework. These levels/modes are expressed as logical policy taxonomy values in Lifecycle Manager and synchronized to Policy Manager for use in physical policy assignment. Such an approach both abstracts operational service policy concepts at development time and greatly reduces the potential for human error in the proxy virtual service generation process. When appropriate, Lifecycle Manager can also be used to automatically apply operational and QoS policies directly to specific service operations.

Beyond straightforward provisioning scenarios, Lifecycle Manager can automate specialized service deployments such as orchestrated services implemented via Policy Manager's process modeling environment and double-hop proxying scenarios where a physical service is virtualized both in the DMZ (to provide first-layer protection against external attacks) and behind the firewall (to provide traditional monitoring and other provider-side policy control over the underlying service). Lifecycle Manager also provides control over runtime listener selection when multiple listeners of the same type (e.g., inbound and outbound HTTP listeners) are defined for a runtime container being managed by Policy Manager.

An integrated Lifecycle Manager / Policy Manager solution can also be configured to reflect rollup operation metrics per service generated by Policy Manager to Lifecycle Manager. This gives the SOA organization's development community full visibility into each service's operational behavior, providing developers with an indication of service capacity and stability and architects with information about the overall health of the organization's SOA.

4.4.3.7 Synchronizing APIs with Community Manager



Lifecycle Manager also automatically synchronizes API definitions with Community Manager, Akana's external API gateway product. This synchronization gives enterprises full visibility over the entire API development and deployment lifecycle, ensuring that configuration settings such as resource-level OAuth provisioning and necessary work products such as documentation, end-user license agreements, avatar images and the like are properly governed and, once approved, automatically provisioned to the external partner community for discovery and consumption. This end-to-end capability is unique to Akana's product suite and provides organizations with an effective way to link business and IT efforts towards building out a set of valuable APIs.

APIs are typically deployed in both sandbox and production modes for access to external consumers. Sandbox API endpoints are commonly provisioned as full production API instances but with test data rather than live data behind the implementation, thereby giving partners (and of course the enterprise itself) an opportunity to validate their app's behavior before going live with the app.

Many customers are repurposing existing SOAP-based services as REST APIs. Lifecycle Manager automation assists in this effort by supporting SOAP-to-REST automated provisioning. When a REST-based API is specified over a SOAP-based service, the owner of the API can establish operation-by-operation mappings from the REST operations to the underlying SOAP operations. Once these mappings are complete, Lifecycle Manager then publishes the API and generates the underlying Policy Manager REST virtual service with the established mappings automatically primed into each of the API's operation process definitions. API developers can then complete the mapping process by specifying content mappings, transforms and other activities within Policy Manager's process editor.

For customers with multi-tenant Community Manager deployments (e.g., for purposes of exposing services as APIs to both internal and external consumers), Lifecycle Manager can be configured with tenant-specific definitions supporting suitable API policy patterns for each target audience.

In addition, Lifecycle Manager can be integrated with Community Manager's App to API access onboarding process to specify configurable onboarding forms and role-based approval processes. The resulting dependencies are automatically populated into Lifecycle Manager and viewable through the Lifecycle Manager UI including via Asset Relationship Visualization.

4.4.3.8 Synchronizing Services with SOLA

Lifecycle Manager provides full service SDLC integration with SOLA, Akana's mainframe service development platform. Both top-down (i.e., WSDL first) and bottom-up (i.e., copybook-based) service

development scenarios are supported via this integration. For the top-down case, the WSDL will be automatically propagated from Lifecycle Manager to SOLA Developer for use during the implementation process. For the bottom-up case, the WSDL generated by SOLA Developer is automatically propagated from SOLA Developer to Lifecycle Manager. In either case, SOLA Developer will notify Lifecycle Manager of development completion, triggering appropriate updates of the SOLA-developed service into Policy Manager.

4.4.3.9 Devops and Cloud Provisioning Integration

As agile and continuous integration-based approaches to software development increasingly take hold throughout the industry, organizations need to be able to manage and control which builds are provisioned to IT operations teams for eventually promotion to production servers. Lifecycle Manager provides prebuilt integrations with Jenkins and TeamCity to support automated identification and approval of such builds designated for promotion. For example, a build might be identified by the development team as a promotion candidate, submitted through a Lifecycle Manager approval process and then automatically provisioned by Lifecycle Manager to one or more target runtime environments for operations-level validation.

In addition, Lifecycle Manager provides the ability to extend this devops scenario with automated provisioning of the target runtime environment via integration with private cloud provisioning platforms. Such platforms are becoming of increasing interests to enterprise IT organizations as they work to reduce operational costs and footprints. By replacing physical testing environments with a virtual server infrastructure combined with cloud provisioning automation tooling, IT organizations can establish on-demand test server provisioning for short or long-term use by the organization as it promotes builds through the operational validation process to production.

4.4.4 Asset Submission Automation Capabilities

As discussed above, Lifecycle Manager provides numerous prebuilt automation capabilities suitable for use during asset submission along with the ability to define custom automation tasks using Beanshell or Jython scripting languages. The following subsections highlight a few of Lifecycle Manager's prebuilt capabilities. For more detail on these and other automation capabilities, refer to the Lifecycle Manager Process Configuration Guide available from the Lifecycle Manager Support Site.

4.4.4.1 XML Validation

Governance processes can be configured to automatically validate XML documents against their imported schema references. These schemas can either be retrieved from an external location referenced by the document being validated, Lifecycle Manager can be configured to provide a local copy of these schemas for validation purposes, or schemas can be directly represented as assets within Lifecycle Manager.

4.4.4.2 Regular Expression Based Asset Validation

Asset version and classifier metadata can be automatically validated using Perl 5 regular expressions (http://search.cpan.org/dist/perl/pod/perlre.pod). Upon detection of a validation failure, an error notification email will be sent as part of a governance process.

4.4.4.3 Policy Manager Based Policy Validation

Lifecycle Manager can be configured to leverage Akana's Policy Manager policy engine for artifact validation. Customers can both use prebuilt policies provided as part of the Policy Manager installation (e.g., WS-I Basic Profile compliance elements) and create their own policies by defining XQuery assertions or Javascript/Java rules. These policies can be automatically invoked as part of a governance process.

4.4.4.4 XML Parsing

If desired, XML documents can be automatically parsed for import dependencies, with discovered dependencies translated into asset relationships (e.g., a BPEL document imports a series of WSDL documents, these WSDL documents in turn import a set of XSD documents; an XPDL document references a series of applications and underlying services). Governance processes can be configured to automatically create assets representing these dependencies if desired.

4.4.4.5 Service Registry Publication

At the end of a successful governance process for a Web service, Lifecycle Manager can be configured to automatically publish the Web service and its associated metadata to one or more targeted service registries. Akana Policy Manager and UDDI-compliant registries such as TIBCO Active Matrix Registry are supported. See section 8.3.1 for additional information on this topic.

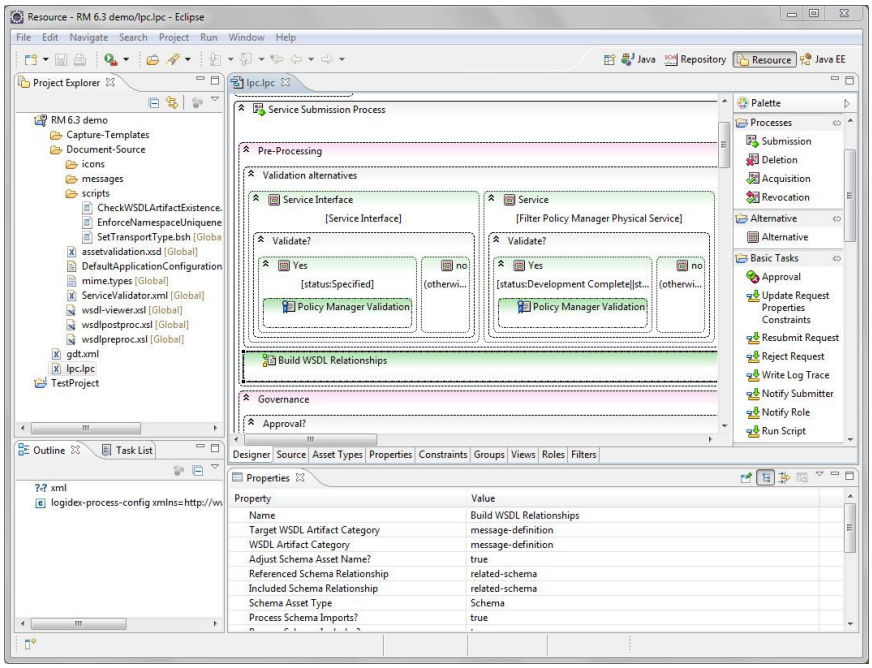
4.4.4.6 Target System of Record Publication

Some organizations may wish to use Lifecycle Manager to validate governed artifacts (i.e., work products) prior to publication into other development systems of record such as SCMs. Lifecycle Manager provides prebuilt automation tasks capable of publishing such artifacts into ClearCase and WebDAV-enabled platforms.

4.4.4.7 Metadata, Template and Process Flow Manipulation

Governance processes can be configured to modify asset metadata and template settings within a configured governance process. For example, it may be desirable to automatically change both the status classifier of an asset and its capture template after publication in order to prepare the asset for the next SDLC stage. Governance processes can be configured to handle special situations such as post-rejection tasks (including automated resubmission if desired) and automatic rejection based on filtered conditions.

4.4.5 Lifecycle Manager Configuration Designer Eclipse Plug-in



To simplify the development of governance automation processes, Lifecycle Manager provides an Eclipse-based plug-in that allows Lifecycle Manager administrators to graphically specify these processes. Using standard palette-based drag-and-drop techniques, administrators can define as many processes as needed to align Lifecycle Manager's automation capabilities with their organization's governance requirements. In addition, Configuration Designer also enables administrators to graphically define asset types (including asset type to capture template mappings), properties, property constraints, metadata and property groups, user roles (including mappings to editable asset types and editable/viewable metadata and property groups) and filters used within role-based asset specifications and governance flows. Administrators can also configure Configuration Designer installations to expose custom-developed automation tasks (known as "listeners") as part of the graphical editing palette. Custom listeners are described further in Chapter 8 of this guide.

Configuration Designer also serves as the primary administrative interface to manage all Lifecycle Manager configuration files. Template files of all types (including HTML-based email notification templates and XML-based asset definition and constraint templates), asset type icon files, asset validation and governance automation XML documents and scripts, and artifact transformation stylesheets are uploaded to and retrieved from the Lifecycle Manager server via Configuration Designer. Configuration Designer can be used to produce a portable zip file representation of a Library's complete set of configuration files for easy server-to-server migration (e.g., from a test environment to a production Lifecycle Manager installation), and can directly import such zip files to produce a new Configuration Designer project within Eclipse or the standalone rich client.

4.5 Users and Their Roles

As discussed earlier, users are assigned roles with the authority to perform certain operations, some of which are Library-wide, while others may pertain only to an organization or project. These roles include:

- Asset User
- Asset Capture Engineer

- Asset Publisher
- Asset Owner
- Project Manager
- Usage Controller
- Library Administrator

Lifecycle Manager administrators can also introduce their own customer-defined roles for use within Lifecycle Manager configurable asset governance processes. These roles can be configured to establish asset and metadata-level scope over creation, editing and viewing content (including state-based edit controls) as well as control over access to specific library capabilities (e.g., actions that should be initiated only by administrators as they result in heavy processing of library content).

4.6 Organizational Groups and Project-level Customization Considerations

Lifecycle Manager supports a flexible mechanism called **Organizational Groups** to model organizations or projects within an IT organization's structure. In general, organizations and projects are populated into Lifecycle Manager based on their involvement in asset production, consumption and/or governance activities and/or the need to establish metrics at various levels of the organizational hierarchy. Group hierarchies can also be introduced to establish asset search visibility rules within the Library (e.g., all Enterprise-scope assets are visible to all Library users, but Group-scope assets are visible only to members of that group and its subgroups). Users can then be assigned on a role-by-role basis to organizations and projects as needed.

Every Lifecycle Manager Library contains an **Enterprise Organizational Group** which represents the entire enterprise (or some other broadly-scoped organization in the event that multiple libraries are deployed within the enterprise). Subsidiary Organizational Groups and Projects can then be created for each initiative as needed. A **project** creates a shared work area for people working on a particular project. Chosen asset views, reference models and asset queries are some of the information managed by the project work area. Projects and Organizational Groups are also useful for SDA management and usage reporting purposes.

Much of the Library behavior is driven by the project definition, which significantly increases productivity and accuracy.

A project defines the following:

- Who is allowed to work on the project.
- Who is the project manager.
- Who is authorized to capture, edit and publish assets.
- What asset views (groups of assets) are visible to the project.
- What Reference Models are visible to the project.
- What the **profile** for the project is that includes:
    - A default asset tree for the display.

- A starting set of classification criteria (for example, a project may only allow components where type = Java to be used).

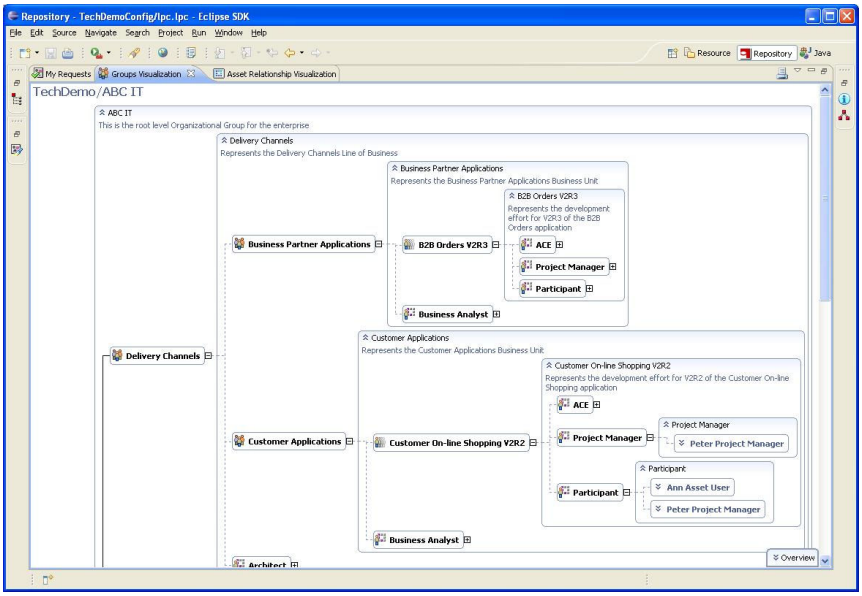
In general, user roles are allocated within the Library's established organizational hierarchy. For example, one user may be allocated the Asset Capture Engineer role for Organizational Group A and another user the same role for Organizational Group B. The first user in this example will be allowed to create and edit only those assets owned by Projects and Organizational Groups contained with Organizational Group A's hierarchy (including of course Organizational Group A itself). Likewise, the second user will be allowed to work with assets owned within the Organizational Group B hierarchy.

This flexible approach to role-based authorization gives organizations the ability to create a layered review and governance model over SDA production. Referring back to the governance example in a prior section of this document, an organization deploying Lifecycle Manager may define the Architect and Business Analyst roles and allocate user authority to those roles at different levels in the deployed organizational hierarchy. For example, the organization may have an Enterprise Architecture team whose members should be involved in review of all candidate assets; thus Lifecycle Manager users with that responsibility will have the Architect role assigned at the Enterprise Group level. On the other hand, Business Analysts may have line-of-business responsibility, so Lifecycle Manager users with Business Analyst review responsibility will have that role assigned to LOB-specific groups within the organizational hierarchy. With this role authorization model, the correct set of asset reviewers will be automatically notified by governance automation based on the group originating the submitted asset.

Administrative roles are established library-wide by default. These roles can also be scoped to users with suborganization-specific (e.g., LOB) responsibilities. Organizations may also choose to automatically provision groups, users and role assignments into Lifecycle Manager from external sources. Such automatic provisioning can occur via integration with LDAP instances or other systems of record via Lifecycle Managers automation APIs. User, project and organization profiles can be extended to include customer-configured properties augmenting the core information managed by Lifecycle Manager.

4.6.1 Groups Visualization

Lifecycle Manager provides an Eclipse-based graphical viewer to visualize your organizational group structure, including the users and their roles distributed throughout the organizational hierarchy. This viewer can simplify the process of maintaining an enterprise's organizational structure within Lifecycle Manager.



4.7 Federated Libraries

Enterprise development groups often need to deal with complex organizational issues, including distributed locations, outsourcing, offshoring, and line-of-business-related partitioning of project teams. Quite often teams need to selectively share SDAs across organizations while maintaining control and oversight over those SDAs. Lifecycle Manager addresses these needs with its federated library infrastructure.

The Lifecycle Manager federated library infrastructure is designed to allow organizations to easily partition their SDA-related activities while supporting selective sharing of SDAs across those partitioned libraries. This multi-library support is enabled at two levels within the Lifecycle Manager product:

- Within a Lifecycle Manager installation, administrators can create as many libraries as desired. By default, each library is totally isolated from all other libraries in the installation. Installation administrators can then selectively connect one library to another, thereby automatically exposing the SDAs resident in the source library (known as the Visible Asset Source) to the client library being configured.
- Administrators can also selectively add libraries hosted by a remote Lifecycle Manager installation to the Visible Asset Source list. These libraries are known as Remote Asset Sources. Lifecycle Manager connects to a Remote Asset Source using SOAP APIs. Once the administrator configures a client library to connect to a Remote Asset Source, that library will refresh its view of the SDAs in the remotely connected library on a periodic basis specified by the administrator.

As assets are published into a local Lifecycle Manager library, their asset metadata is applied against filtering criteria established for connected libraries. Such filters are configured using the Configuration Designer discussed earlier in this document, and can combine multiple metadata elements and values as needed to enable precise control over which assets are to be synchronized over a remote connection (e.g., only enterprise-certified services are to be shared across libraries). If the asset conforms to filtering criteria, its metadata will flow to the connected library and become visible within that library. Of course, this library in turn may be connected to further downstream libraries allowing organizations to implement and deploy quite sophisticated asset distribution models.

In an integrated Lifecycle Manager / Policy Manager deployment, services federated across libraries can be acquired by applications or services in any library where they are visible. Lifecycle Manager automatically maintains service ownership across all libraries in a federated configuration, thereby allowing it to automatically route consumed asset approvals to the correct remote library (e.g., routing an application's request to use a service with a specific service-level agreement to the owner of that service in its originating library) and to establish operational contracts to the correct service instances managed by Policy Manager. See section 6.4 for further discussion of integrated service acquisition.

4.8 Key Library Configuration Points

While many aspects of the Lifecycle Manager work environment are defined at the project level, certain aspects are specified at an "enterprise level" affecting the entire Lifecycle Manager installation.

Lifecycle Manager provides some configuration points to help manage the installation and support of the Library for multiple groups, including:

- Library description
- "Sender Email Account" for automatically generated messages – to receive replies
- Feedback contact
- What information will be displayed on asset information pages
- Global Template (default) definition (see templates discussion)
- Asset acquisition mode(s) (see chapter 6 for details)

4.9 The Quickest Way to Get Started

Lifecycle Manager is very flexible, but ships with default settings, preconfigured schemas and governance automation flows, and standard templates to speed start-up. The following tasks will also make start up faster:

- Work with a small set of assets to start.
- Identify a small set of users to start.
- Import a simple, existing UML or block-diagram based model which covers the domain or part of the domain of the assets to be loaded (load the model used to create the assets, if it is available).
- Capture assets using the IDE-based wizard or web submission interface, saving automatic updating and loading for later.
- Apply appropriate governance process automation to ensure published assets are of good quality and meet organizational needs and expectations.
- Define a minimal organizational structure sufficient to represent project collaboration and governance scoping requirements.
- Initially, have the same person be the Library usage controller and administrator.

These and many other recommendations are detailed further within the Lifecycle Manager Best Practices and Assessment template documents.

5 Locating and Using Assets

The core value of the asset Library is its ability to allow users to locate and understand assets that fit their specific requirements. A properly configured Library will result in effective asset understanding and use.

Lifecycle Manager provides several search mechanisms which support all styles of search interaction, ranging from a quick ad hoc keyword search to a sophisticated search using Reference Models.

Lifecycle Manager supports both thin-client (browser-based) and IDE-based user interfaces. Both Internet Explorer and Firefox browsers are supported. Lifecycle Manager provides IDE integrations for Eclipse-based IDEs (both open source and commercial variants) and Microsoft Visual Studio.

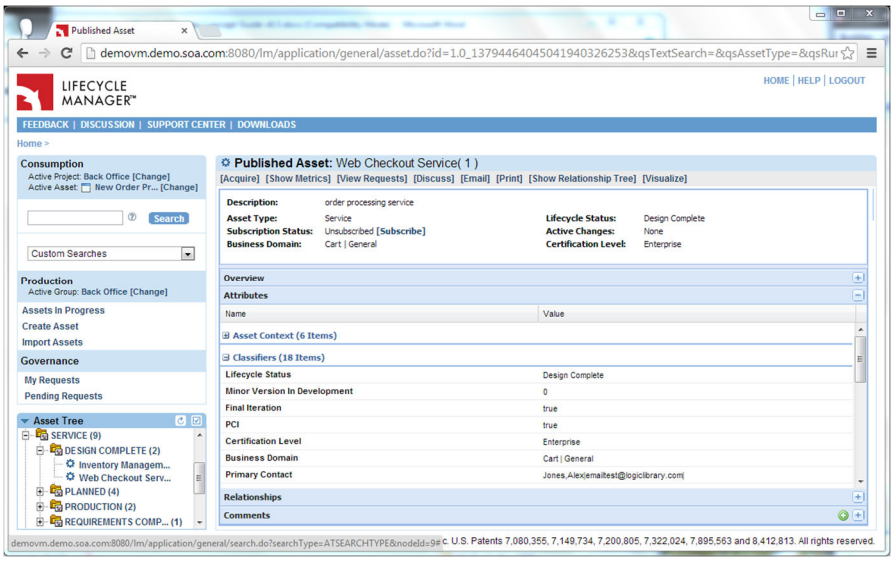
Locating Assets

- Asset Tree
- Browsing Models
- Ad hoc Search
- Asset Query
- Model Based Search
- IDE Integrations
- Asset Viewer

5.1 Modes of Locating Assets

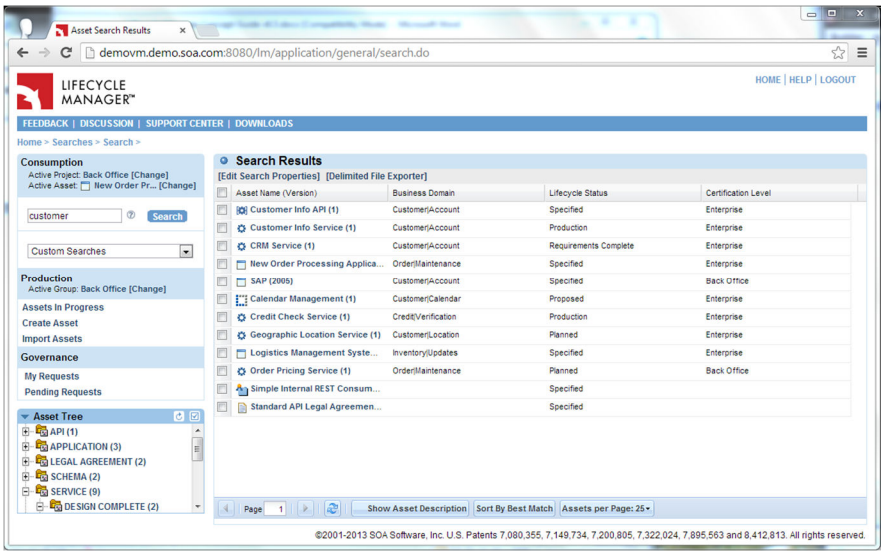
5.1.1 Browsing for Assets

A user can browse for assets using the Lifecycle Manager Asset Tree, which is always present on the left hand side of the Lifecycle Manager user interface. The Asset Tree is similar to an explorer tree or a file tree but is dynamically generated from project- or user-selected classifier types. The tree can be navigated, and when an asset is chosen, its detail is displayed.



5.1.2 Searching For Assets

Searching for assets allows a user to find assets associated with specific criteria, such as application requirements. A search can either be persistent or ad hoc and can be either text-based or based on a graphical reference model previously populated into Lifecycle Manager. Search results then return a list of assets conforming to the search criteria along with up to three columns of configured classification information that can be used to sort the assets prior to further investigation.

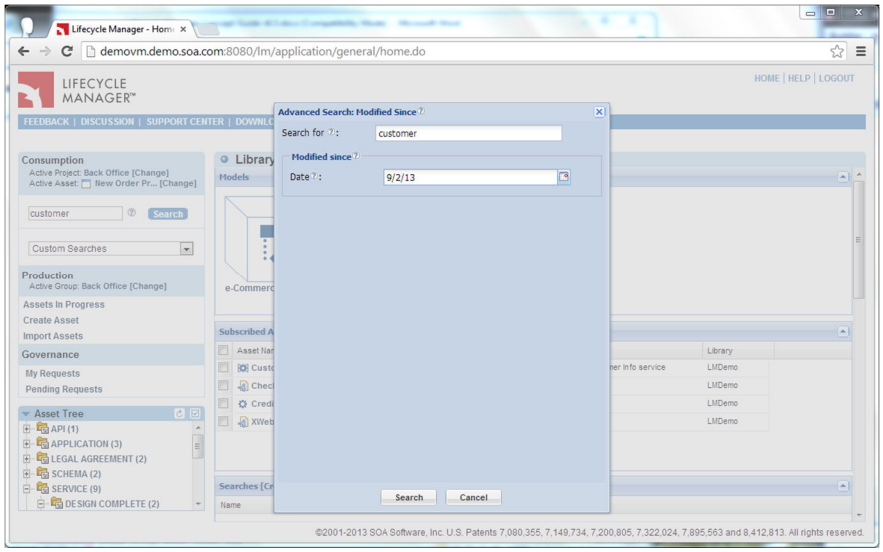


Lifecycle Manager can aid in the control and collaboration efforts of locating assets. Lifecycle Manager has the concept of a **project**. A project can be thought of as a shared work area which ensures that assets used in the project conform to a given **profile**. In addition, projects provide an area for searches to be stored, used, and/or reviewed by others. (See other aspects of projects in Section 4.6, Project Level Customization Considerations and Section 6.4, Acquiring Assets.)

5.2 The Asset Tree

The asset tree is a very flexible and intuitive mechanism for locating assets. It operates much like a standard file search and exploring mechanism, yet its structure is dynamic and can change based on categorization values – such as asset type, asset domain, provider, etc. Users can also modify the structure of the Asset Tree based on their own preferences, or use the Asset Tree structure provided for them by the project they are working with. Asset Tree nodes can be used simply to organize assets within the hierarchical tree view, or can be directly invoked to build a search result list of all assets that reside under that node. Users can configure their Asset Tree settings to automatically expand to the desired depth upon login if desired.

5.3 Preconfigured and Custom Searches



In Lifecycle Manager, a standard search box is always visible in the left-nav bar of the Library. Searches can be initiated either from this left-nav search box or directly from the asset editing page while adding asset relationships. This latter approach is particularly useful to asset owners as they progress an asset through its defined governance lifecycle. In many cases, specific relationships must be established by the asset owner at each governance stage, and by allowing owners to initiate search within asset edit Lifecycle Manager allows those users to complete this work while in the broader context of fulfilling the content mandates established for the current governance stage of the asset under edit.

The user can select any of the following types of preconfigured searches:

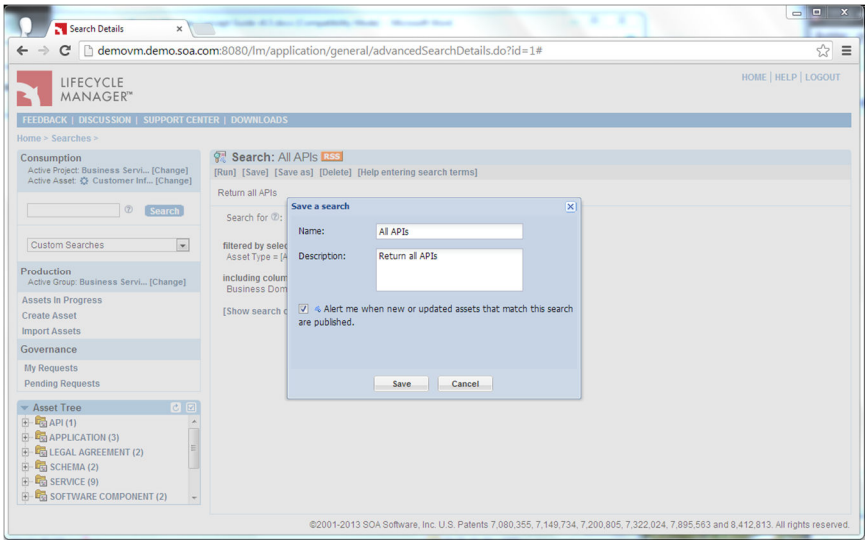
- **Assets by Content**, which searches across all available asset metadata and queryable artifacts. If no search string is entered, the result will list all assets that are available in the active project.
- **Assets by Name**, which searches for a given asset by name.
- **Assets by Keyword**, which searches for assets with a given keyword.
- **Services by Operation**, which searches for operation names specified within WSDL artifacts associated with service assets
- **Assets by Status**, which combines search for asset content with an additional search field based on asset governance status
- **Modified Since**, which combines search for asset content with a date field used to filter out those assets whose last modification date is earlier than the provided date value

In addition to these standard search types, Lifecycle Manager administrators can also configure custom search types leveraging the product's XML-based search engine. Every asset managed by Lifecycle Manager along with its surrounding context (e.g., owning organizational group, last editor, creation date, last edited date) is stored as an XML document that can be searched via configured XPath expressions. Lifecycle Manager administrators can declare as many XPath expressions as desired and expose them to end users as search types alongside Lifecycle Manager's standard searches. These XPath expressions can also be extended to incorporate any XML-based artifacts associated with Lifecycle Manager assets (e.g., BPEL, WSDL, XSD). By configuring specialized search types, administrators provide Lifecycle Manager users the benefit of precise user-centric searches without requiring those users to understand the complexities of XPath.

As is evidenced by the Assets by Status and Modified Since default search types, Lifecycle Manager supports multi-field searches. Configured search expressions can be combined as desired to create compound search forms of any desired type, giving Library administrators the flexibility to define sophisticated search types that meet the needs of various Library audiences in a simple, easy-to-understand manner.

Once an asset search is executed, a list of assets meeting the criteria is displayed along with a configurable and sortable set of metadata for these assets. These assets can then be investigated and deployed. Adhoc searches can also be saved as persistent searches for further refinement.

5.3.1 Saved Searches



Saved searches are persistent records of search criteria. These records hold the definition of an asset query. Saved searches are owned by and stored in a project and can be accessed by any authorized project user. Any search initiated from the search box can be saved by selecting the Edit Search Properties action on the Search Results page. This takes the user to the Search Details page where search criteria can be modified and the search named and saved for future use.

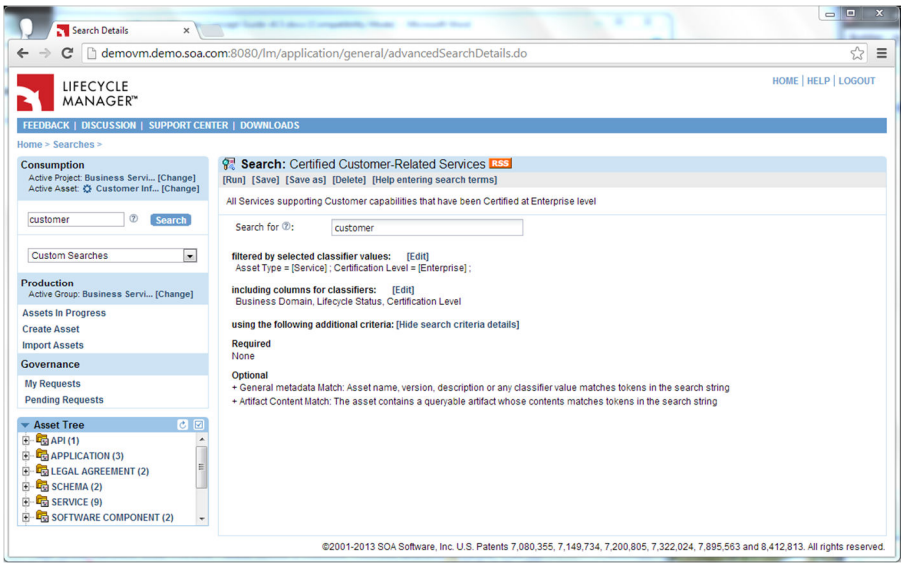
Users can mark any saved search as "**alertable**", thereby enabling those users to be notified of any new or republished asset that conforms to the search. This powerful alert mechanism helps users to automatically stay on top of new information (such as architectural patterns and best practices), improving communication within the development organization.

Saved searches offer the ability to:

- Rerun the same search at any time to update the resulting asset list.
- Modify the search criteria and rerun the search.
- Make the search criteria available to others

5.3.2 Searching with an Asset Query

Searches can also be created directly from the Lifecycle Manager's home page by selecting the Create Asset Query action. Searches created in this manner are identical to those created through the search box; however, this approach also allows the user to immediately enter classifier filtering criteria to narrow the search.

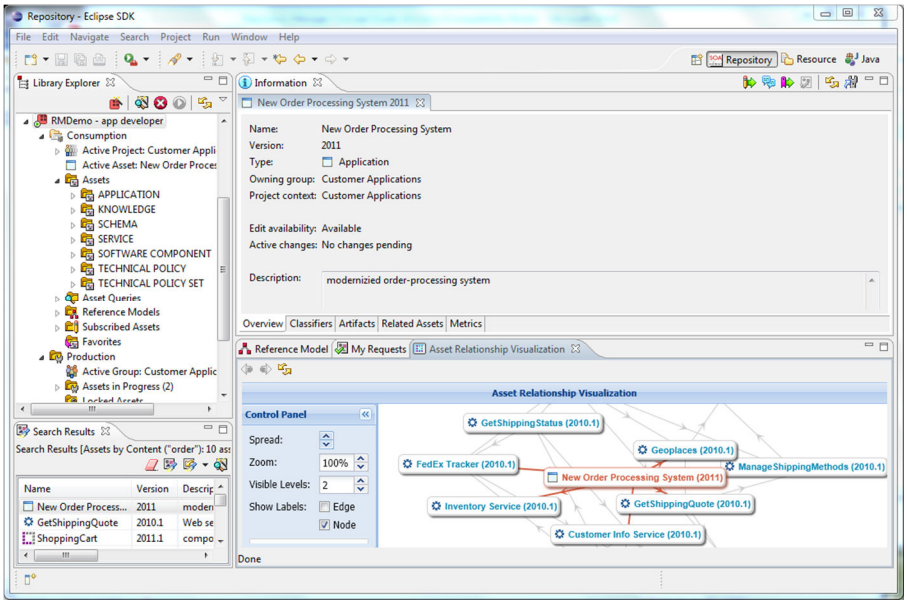


5.3.3 Search Results

Regardless of the method used to invoke the search, Lifecycle Manager returns the set of assets confirming to the search in a configurable and flexible Search Results grid. This grid lists each asset's name and version along with up to three designated classifiers in a sortable and paginated format. Designated classifiers are specified by organizational profile and can be overridden by individual user settings as desired. Depending upon how the search was initiated, this grid may be presented to the user as a new Library page or may be presented as an asset edit dialog.

When initiating a search from the left-nav search box, users can choose to export all or some of the assets returned from a search. When export is selected, Lifecycle Manager produces a zip file containing a spreadsheet listing all metadata for the selected assets and a series of asset-specific folders containing any by-value artifacts associated with the exported assets. This export result set can in turn be modified and used to update assets via Lifecycle Manager's Import Center (section 8.3.2), or simply manipulated by the user to produce ad-hoc reports using spreadsheet or other external tool capabilities. Depending upon Library configuration, other search results processing actions may be available from the Search Results page; for example, a user responsible for creating project proposal assets within the Library may initiate an asset search, investigate the assets returned by the search and finally select one or more of those assets to be automatically associated with a newly generated Project asset initiated from the Search Results page.

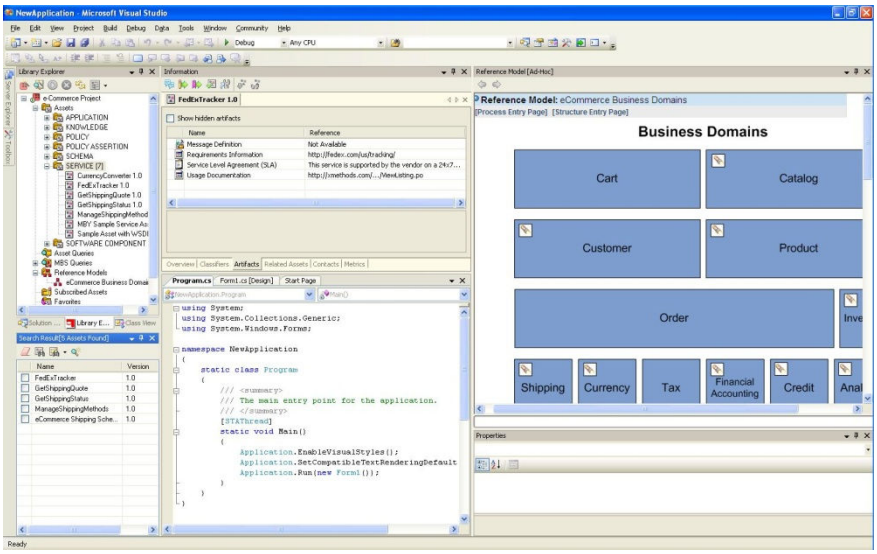
5.4 RAD and Eclipse Plug-ins



In addition to the thin-client user interface shown in the previous sections, Lifecycle Manager provides a powerful, tightly-integrated plug-in to IBM Rational Application Developer and other Eclipse-based IDEs. These plug-ins give developers all the capabilities of the Lifecycle Manager thin-client but directly accessible right from their IDE, eliminating the need to move from one tool to another during development.

Because the Lifecycle Manager plug-ins are tightly integrated, users can easily configure them to support their own usage patterns and activities. All of the actions supported by the plug-ins use standard RAD/Eclipse widgets and wizard structures. In addition, ClearCase users can import ClearCase controlled artifacts by reference directly into their ClearCase controlled Eclipse projects via the Lifecycle Manager plug-in, thereby eliminating the copying and duplication of executable artifacts such as jar files and ensuring ongoing synchronization with the correct file version. Supported ClearCase options include both UCM mode and non-UCM mode and Composite Component references.

5.5 Visual Studio Add-in



Lifecycle Manager also provides a powerful, tightly-integrated add-in to Microsoft's Visual Studio IDE. This add-in gives developers all the capabilities of the Lifecycle Manager thin-client but directly accessible right from their IDE, eliminating the need to move from one tool to another during development.

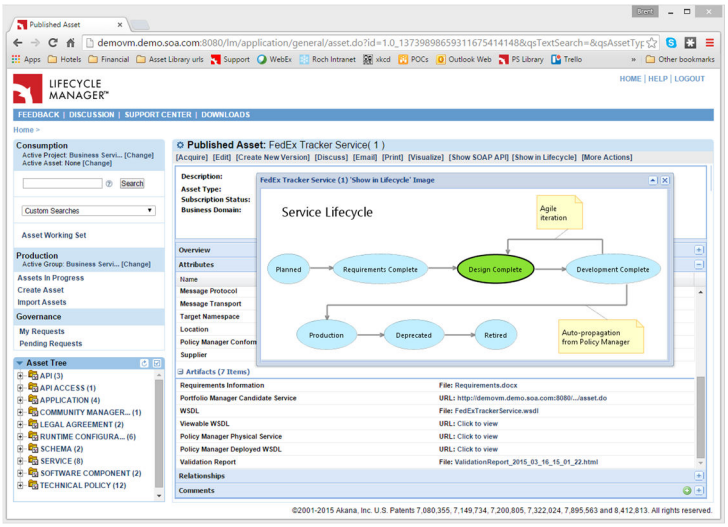
Because the Lifecycle Manager add-in is tightly integrated, users can easily configure it to support their own usage patterns and activities. All of the actions supported by the add-in use standard Visual Studio widgets and wizard structures.

5.6 Other IDEs

Lifecycle Manager's Eclipse Plug-in is compatible with many other commercial Eclipse-based IDEs. Contact Akana support for questions about specific IDE products.

6 Employing Assets

6.1 The Total Asset



Lifecycle Manager holds the overview, metadata (classifiers), properties, artifacts, and related assets – all of which make up the "total view" of an asset.

Examples of the information and references Lifecycle Manager holds for each SDA include:

- **Overview Information**: name, version, overview, description, contacts and supporting project information.
- **Attributes**: classification information such as asset type, supported platforms, functional domain & sub-domain, support level, and other metadata elements configured into the Library for the asset type, along with artifacts (i.e., work products) such as
- XML documents, binaries, user documentation, test scripts, models, and others.
- **Asset Relationships**: previous versions, prerequisites, co-requisites, suggested/actual usage and other strongly-typed dependencies between assets.
- **Context Information**: what group owns the asset, when was it created, when was it last modified, etc.
- **Lifecycle State**: a visual representation of the asset's current SDLC state
- **Governance Request History**: access to all past submission requests for the asset as it progresses through its production SDLC.

Each asset type is associated with a unique icon providing a useful visual cue to Library users. Asset type icons are shown in the Asset Tree, Asset Details (for both the asset being viewed and its related assets) and Asset Relationship Visualization (ARV) views. Default icons are provided for each asset type provided with the Library's default configuration. These icons can be replaced as desired by customers, and additional icons can be provided for customer-specified asset types.

Lifecycle Manager can be configured via XML templates to define the exact information to be stored for each given SDA type (see Chapter 4: Building the Library and Integrating a Development Environment). Hints are also provided to the user to indicate if the asset currently in view has a change in process. Knowledge of such a planned change may cause the user to contact the asset owner to understand if the change will affect the planned use of the asset.

6.1.1 ***Virtual Asset Window™***

Different roles in an enterprise require different information concerning an asset to facilitate understanding, governance and consumption of that asset. For example:

- Business analysts need information about how the asset supports business requirements
- Architects need to see how the asset fits with the preferred architectural approach
- Project managers need to understand schedule, effort, and potential ROI of the asset

Employing Assets

- Investigate
- Discuss
- Visualize
- Acquire
- Subscribe
- Cross-tool IDE integrations

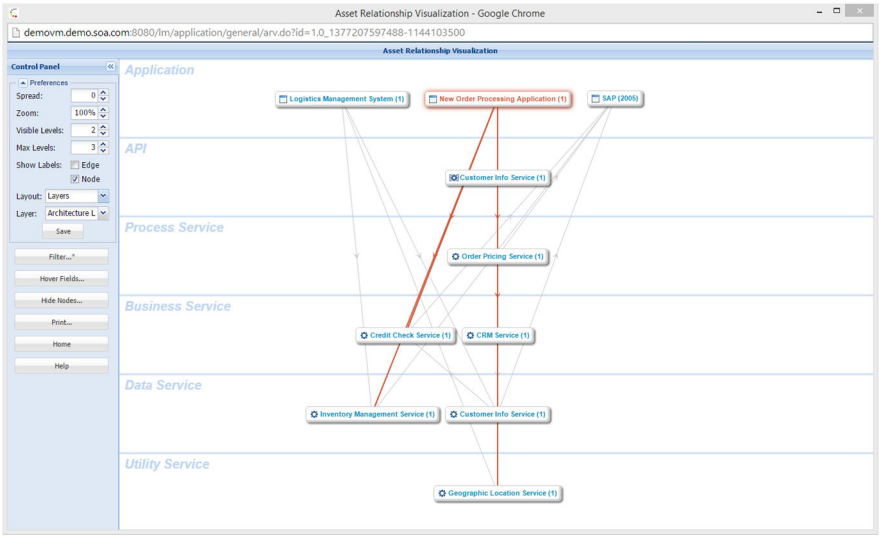
To drive improved understanding of assets/services, Lifecycle Manager's Virtual Asset Window extends role definition capabilities to the asset presentation level. For each defined role within Lifecycle Manager, an asset can be configured to present the asset in a role-specific fashion with respect to the information exposed and how that information is ordered on the user interface. In addition, users with multiple roles can easily switch from one role to another as they view an asset. With the addition of the Virtual Asset Window, users can quickly and easily see only the information they need concerning an asset, thereby broadening the types of users accessing, using and promoting assets throughout the enterprise.

6.2 Studying and Discussing Software Development Assets

The user can review the information available for any asset that is visible to them, and even launch or download asset workproducts as desired. For service assets, Lifecycle Manager is configured by default to automatically populate a WSDL viewer artifact that presents WSDL contents in an easy-to-read human-friendly format. Discussion threads (including an optional inline Comments section presented directly on the thin-client Asset Detail page) are also available for each asset that can be reached from any of the asset information pages. Users can review these discussions or join them. Users can also choose to email asset information to others directly from the thin-client Asset Detail page.

If an older or newer version of the asset being reviewed exists in the Library, users can view changes between versions in digest form through the Lifecycle Manager thin client UI. In addition, WSDL and other XML document-type artifacts within such versioned assets can be compared using Lifecycle Manager's built-in artifact comparators. Lifecycle Manager also allows customers to implement and deploy their own custom comparators for these and other artifact types as desired.

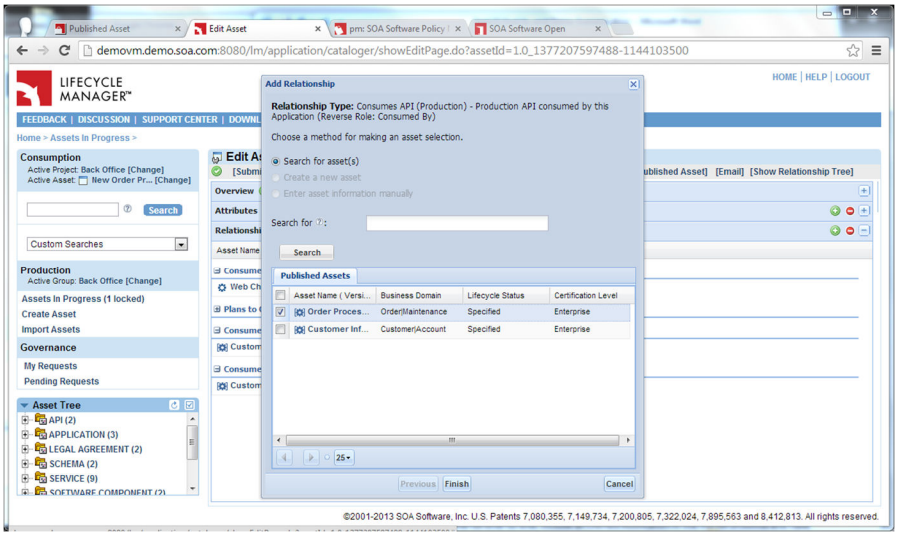
6.3 Visualizing Assets and their Relationships



Lifecycle Manager users can choose to view a selected asset in relationship to other assets using Lifecycle Manager's Asset Relationship Visualization (ARV) capabilities. ARV provides a Relationship Visualization view that presents a graphical overview of multi-level dependencies centered on a selected asset. Both radial and layered views are supported. ARV users can view asset and relationship content digests via hoverhelp, and they can choose to launch any asset within the ARV view into the main Library window as desired. Likewise, ARV views are fully supported within Lifecycle Manager's Eclipse and Visual Studio plug-ins.

User-configurable viewing options for ARV include filtering on one or more relationship types, asset metadata and selectable recursion depth. Users can also select specific metadata elements to expose in asset hoverhelp, and can individually hide specific assets to produce the desired graph layout that meets user objectives. ARV settings can be named and saved for reuse.

6.4 Acquiring Assets for Use

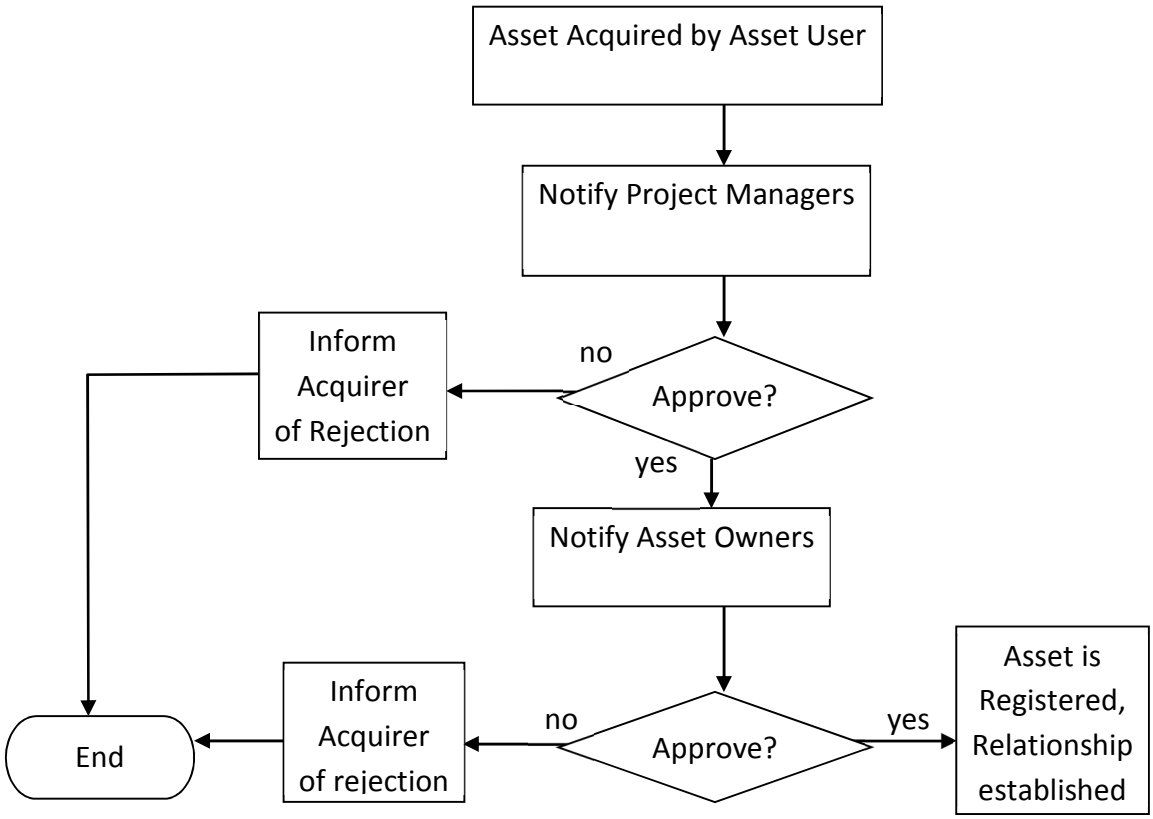


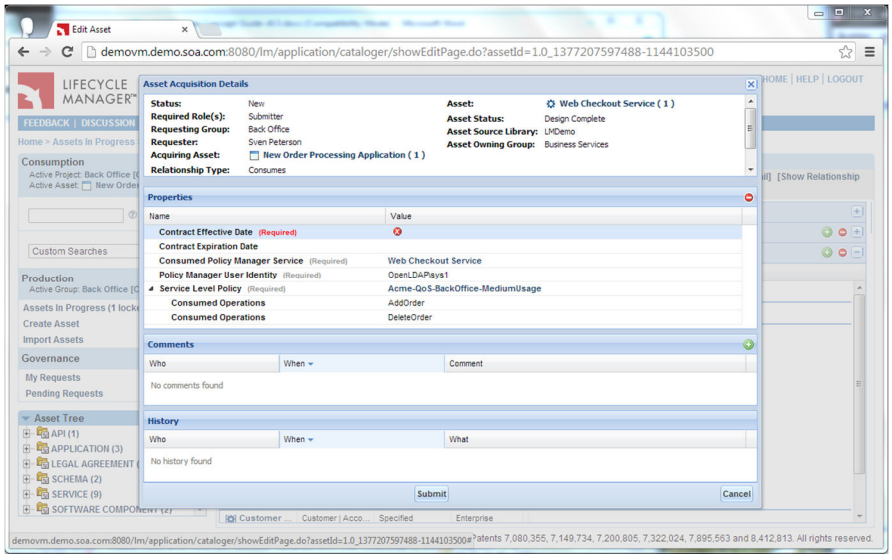
Once an asset editor has determined to consume an asset, he or she clicks the acquire icon/link in the IDE plug-in/browser to initiate asset acquisition. Alternatively, selection of an asset via the Add Relationship wizard when editing an asset in the browser automatically initiates the asset acquisition process. Asset users can easily determine the project and consuming asset currently in force by viewing the Active Project and Active Asset fields within Consumption section of the left-nav bar of the thin-client UI or by viewing the equivalent entries within the Connection node within the plug-in's Library Explorer view. Upon initiation of an acquisition, an asset user may be asked to provide justification for or more details about

the acquisition (e.g., the quality of service required by an application's use of a service). This justification may be as simple as a free-form text comment, or may involve specifying expected service-level demands to be placed on the asset, agreement to corporate usage criteria, or other information needed to evaluate or audit asset usage. When Lifecycle Manager is integrated with Akana's Policy Manager, the decisions made during asset-based acquisitions of services result in enforceable runtime contracts being automatically generated by Lifecycle Manager into Policy Manager. These contracts can be established at the service operation level if desired, giving service consumers full control over quality of service requirements on an operation-by-operation basis.

Just as for asset production, governance processes can be configured to control and automate the asset acquisition process. For many organizations, asset acquisitions will require approval, perhaps by a Project Manager or by a representative of the organization that produced the asset being acquired (e.g., the owner of a service must approve all requested service usage to ensure that the service's production infrastructure is not overwhelmed by consumer demand). Lifecycle Manager governance automation provides a fully flexible acquisition governance model, supporting customer-defined roles, automation and manual review stages with configurable checklist-style fields and metadata-driven process filtering. Out of the box, Lifecycle Manager is configured to enable a two-stage acquisition governance process involving the Project Manager and Asset Owner native Lifecycle Manager roles. If the project has been configured to require a Project Manager's approval, this request is put into the Project Manager's acquisitions queue and all managers for that project are notified via email. Because asset acquisition may result in cross-charging, license purchases or usage certification, or other aspects of project governance, this approval process ensures that the project manager agrees with the use of the asset. Likewise, if Lifecycle Manager has been configured to require that the Asset Owner approve all asset acquisitions, then the request is put in the owner's acquisitions queue allowing the owner to approve or deny the request. As noted in the Federated Libraries section above, Asset Owner approval extends to remotely-owned assets in a federated library configuration, thus ensuring that proper authority is granted to cross-enterprise consumers while preserving library-scoped security.

The default acquisition governance process is graphically portrayed as follows:





Lifecycle Manager's prebuilt integration with Akana Policy Manager provides a powerful mechanism to coordinate the development-time and runtime views of service consumption contracts. Valid Service Level Policies (SLPs) defined within Policy Manager for the service being acquired are directly exposed to the developer initiating the acquisition, thereby allowing the developer to document the intended usage characteristics for the service client (e.g., application, orchestration) currently under development. In addition, if Network Director or IBM DataPower is being used to establish proxy virtual services for consumption purposes, Lifecycle Manager can present the full set of these proxy services with their varying non-functional characteristics to the developer initiating the consumption request. If desired, integrated consumption governance processes can also be configured to allow operation-level agreements to be established between the consumer and the consumed service. When the consumer is also a service, each operation

of the consuming service can establish its own independent consumption agreement with one or more operations of the consumed service, thereby enabling fine-grained impact analysis visibility and control over service access. Upon approval, this information is used to automatically create a Policy Manager contract (or contracts in the case of operation-level acquisition) in draft state that serves as the basis for operational governance. These contracts typically include reference to the consumer's unique identity – a UDDI tModel key in the case of a consuming service, or an authentication identity such as LDAP userid in the case of a non-service consumer. Contracts can also be scoped to the consuming project if desired.

Lifecycle Manager acquisition processes can also be configured to automatically acquire related assets upon approval of a root asset. For example, an IT organization may have established a two-tier service architecture layer composed of composite and elemental services, where composite services incorporate one or more elemental services within their implementations. It may be beneficial in this environment to track acquisitions of not only composite services by applications but also the underlying elemental services implicitly consumed by those applications through composite service use. Related asset acquisition can be configured to acquire only directly related assets or to recursively acquire assets of specified relationship types.

Users involved in the asset acquisition approval process are automatically notified of pending requests via an email which contains an imbedded URL to the request details. Reviewers can also access via the Lifecycle Manager thin client a list of pending requests as well as those requests that they have acted upon in the past. Users with Usage Controller authority are given access to an additional page which lists all requests in the library. All of these pages provide appropriate sorting and filtering options to allow the user to quickly find the desired request based on submitter, status, request type, and other parameters.

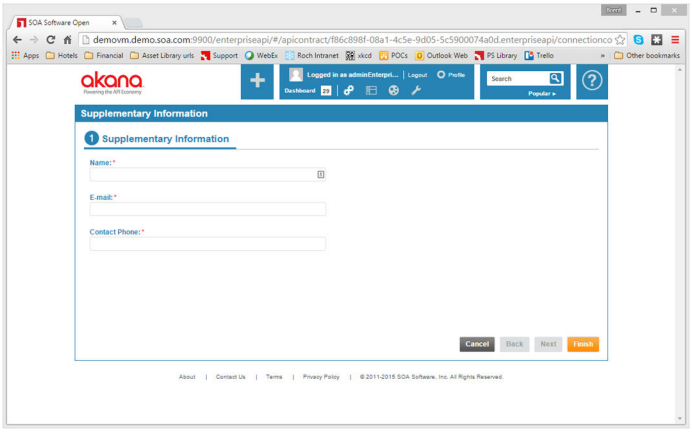
Asset acquirers can view their active and previously processed requests through a similar set of pages in the Lifecycle Manager thin client. Active requests show acquirers the currently pending governance approval role along with Lifecycle Manager users having that role-based authority to approve or reject the acquisition request, thereby making it easy to ping approvers for requests that are not being expeditiously processed. Acquirers needing to resubmit an asset (e.g., because a role-based reviewer rejected the original acquisition request) can choose to prime their updated acquisition request with request field values from the immediately prior acquisition request for that asset.

Depending upon the sensitivity of the asset being acquired, some asset publishers may choose to configure certain asset artifacts (such as binary executables, detailed design documentation, etc.) as **private**. When an artifact is configured as private, it cannot be accessed through the Lifecycle Manager Library until all asset acquisition approvals have been successfully completed. This is a useful feature for organizations that want to maintain tighter control over asset usage and deployment.

6.4.1.1 Lifecycle Manager Configuration Designer Eclipse Plug-in

As discussed in section 4.4.3.1, Lifecycle Manager provides an Eclipse-based plug-in to support graphical definition of governance automation processes.

6.4.2 Portal-based API Acquisition



When integrated with Community Manager, Lifecycle Manager supports a fully configurable App to API access approval process similar to the capabilities provided by Lifecycle Manager's native UI and IDE-based integrations. Lifecycle Manager provisions not only the core API definition to Community Manager but also "full product" configuration elements such as shared legal agreements, operation-level licenses and extended API documentation. Using Lifecycle Manager's REST-based Lifecycle APIs, Community Manager administrators can configure specialized onboarding forms and role-based approval processes to support the technical, business, legal and regulatory needs of the enterprise via a fully automated and auditable workflow.

6.4.3 Active Asset Analysis

Some organizations may desire a formalized feedback mechanism (beyond the asset-specific discussion forums automatically created by Lifecycle Manager for each asset discussed in section 6.2 above) for acquired assets. Once an asset has been acquired by a Lifecycle Manager project, Lifecycle Manager can be configured to automatically generate a user feedback survey for that asset. This survey is configurable and can contain both quantitative (e.g., how much effort was saved) and qualitative (e.g., how did you use the asset) feedback in both numeric and text form. Upon submission of the completed survey by the asset user, Lifecycle Manager automatically updates the rated asset and its discussion topics designated for textual feedback, with the survey response preserved for audit and/or historical purposes. Such formalized consumption feedback concerning how well an asset has met the needs of its consumers can be a critical aspect of establishing, maintaining and increasing asset reuse throughout the enterprise. Also, consumption feedback can guide an organization on identifying areas where the asset lifecycle, governance and management practices can be expanded and improved.

6.4.4 Integrated Asset and Project Usage Metrics

Lifecycle Manager has integrated Eclipse's Business Intelligence and Reporting Toolkit (BIRT) runtime engine as part of its server deployment. As part of this integration, Lifecycle Manager automatically generates usage metrics-oriented reports for every asset, project, and organizational group in a Lifecycle Manager library. These reports show the list of all acquisitions (including operation-level usage dependencies for APIs and services) and subscriptions for each asset, and the list of owned assets and their acquisitions for each organizational group (the project metrics report also shows the list of assets acquired by the project), giving Lifecycle Manager users easy access to focused information about library activity levels at a granular level.

6.4.5 ***Envision Integration***

Lifecycle Manager provides out-of-the-box integration with Envision, Akana's analytics platform. The following two Envision connectors feed relevant asset and governance information to Envision for exposure in Envision's dashboard:

- Service Lifecycle Velocity: a graph which shows average transition times for the organization's services as they progress through the organization's SDLC gates with group-specific drilldown
- Service Heat Map: a pie chart which displays service counts organized into the organization's business domains with drilldown to subdomains and lower levels as specified in Lifecycle Manager's configuration

Additional connectors can be implemented by Akana professional services or customers as desired to provide additional visibility into various aspects of the development process.

6.5 Subscribing to Assets to Keep Informed

If a user is considering an asset for acquisition, or works with an asset on an on-going basis (such as design standards or core central services), he or she can subscribe to that asset for information about its changes. Subscribers receive email notification when an asset has changed or a new version is published.

Users can subscribe by clicking on the subscribe link on the Asset Detail Page or the subscribe icon within the IDE plug-in. Each user can manage his or her list of subscribed assets via the Library home page.

Users can also choose to subscribe to asset forums via RSS feed. Since every asset republication results in a new posting to the Publication Notes topic for the asset, this RSS-based approach provides an alternative to email-based asset subscriptions for those users that prefer RSS aggregator/reader-style notifications.

7 Assessing – An On-going Process

7.1 Reporting on the Effectiveness of the Library

Ongoing Assessments

- Attainable, Measurable Goals
- Regular Analysis and Reporting
- Ongoing Planning and Roll Out

It is important to track metrics associated with goals set for the Library in the initial assessment and planning phase. Metrics might include:

- Number of Assets Reused (acquired)
- Cost saved via Reuse
- Number of Assets in the Library
- Top Asset Authors (as indicated by most used assets)
- Library Users
- Discussion forum activity

Akana's professional services can help determine the appropriate metrics for a company's needs. Professional services can also help design reports and custom templates for gathering and analyzing data about Lifecycle Manager.

7.2 Reporting Views

Lifecycle Manager collects the information needed to support reporting and analysis requirements. The information is available through read-only views into the production database, which supports the use of standard report writers and information warehouse/mining tools. The views are designed for easy access and understanding.

Data Views include:

- Assets in Lifecycle Manager Catalog (RPT_CATALOG_ASSET)
- Published Assets (RPT_ASSET)
- Asset Artifact Information (RPT_ASSET_ARTIFACT_INFO)
- Asset Classifier (RPT_CLASSIFIERS)
- Asset Forum Information (RPT_ASSET_FORUM)
- Asset Forum Topics (RPT_ASSET_FORUM_TOPIC)
- Asset Forum Messages (RPT_ASSET_FORUM_MESSAGES)
- Asset Acquisition Request (RPT_ASSET_ACQ_REQ)
- Asset Registrations (RPT_ASSET_REGISTRATION)
- Asset Changes (RPT_ASSET_CHANGE)
- Asset Classifier (RPT_ASSET_CLASSIFIER)
- Asset Relationship (RPT_ASSET_RELATIONSHIP)
- Asset Function (RPT_ASSET_FUNCTION)
- Manual Publish Requests for New Assets (RPT_ASSET_PUB_REQ)
- Asset Requests (RPT_ASSET_REQUEST)
- Asset Request Actions (RPT_ASSET_REQUEST_ACTION)
- Asset Request Approvals (RPT_ASSET_REQUEST_APPROVAL)
- Asset Subscriptions (RPT_ASSET_SUBSCRIPTION)

- User Query Alert Subscriptions (RPT_QUERY_SUBSCRIPTION)
- Asset Search Specifications (RPT_SEARCH_SPECIFICATION)
- Asset Sources (RPT_ASSET_SOURCE)
- Domain Reference Models (RPT_DRM)
- Assets Mapped to Reference Model (RPT_FUNCTION_MAPPING_INFO)
- Reference Model Idealized Components (RPT_IC)
- Reference Model Idealized Component Functions (RPT_IC_FUNCTION)
- Reference Model Idealized Component Interfaces (RPT_IC_INTERFACE)
- Asset Global Definition Template (RPT_GLOBAL_DEF_TEMPLATE)
- Asset Capture Templates (RPT_CAPTURE_TEMPLATE)
- Library (for use in including Library name in report) (RPT_LIBRARY)
- General Forum Information (RPT_GENERAL_FORUM)
- General Forum Messages (RPT_GENERAL_FORUM_MESSAGES)
- General Forum Topics (RPT_GENERAL_FORUM_TOPIC)
- Organizational Groups (RPT_ORG_GROUP)
- Organizational Group Hierarchy (RPT_ORG_GROUP_HIERARCHY)
- User Role (RPT_ORG_GROUP_USER_ROLE)
- Projects (RPT_PROJECT)
- Project Users (RPT_PROJECT_PARTICIPANT)
- User (RPT_USER)
- Contacts (RPT_CONTACT)
- Audit Trail Events (RPT_AUDITTRAIL)
- Audit Trail Event Properties (RPT_AUDITTRAIL_PROPERTIES)
- Audit Trail Events and Event Properties (RPT_AUDITTRAIL_COMBINED)
- Lifecycle Manager Entity Properties (RPT_PROPERTY)
- Asset Request Properties (RPT_ASSET_REQUEST_PROPERTY)

7.3 Integrated Reports and the Eclipse BIRT Reporting Engine

Lifecycle Manager ships with integrated Eclipse Business Intelligence and Reporting Toolkit (BIRT) support. BIRT is an open source, Eclipse-based reporting system that integrates with applications to produce compelling reports for both web and PDF. BIRT provides an Eclipse-based design time toolkit along with a server-side runtime engine. BIRT's capabilities are a natural complement to Lifecycle Manager, allowing Lifecycle Manager users to rapidly develop new reports in a graphical rich client environment and easily deploy those reports to the integrated BIRT server engine within Lifecycle Manager.

Lifecycle Manager also automatically generates the following BIRT-based library-wide reports for every library created within a Lifecycle Manager installation:

- Business Domain Heatmap Report
- Success Metrics Report
- Repository Savings Report
- What's Hot Report
- What's New Report
- Search Analysis Report
- Governance Monitoring Report
- Stale Assets Report
- User Assignments Report

Additional details about these reports are available by downloading the Lifecycle Manager Reports Toolkit from the Lifecycle Manager Service Center.

8 Extending Lifecycle Manager to Project Management and Deployment Environments

8.1 Project Management, Development and Deployment

As enterprises move forward with component-based and service-oriented application architectures, they increasingly recognize the need to better tie the developed components and services with their deployed instances, both to improve the control and efficiency of application deployment and maintenance and to provide rapid and meaningful feedback to their developers when things go wrong. In addition, many enterprises are looking to increase their automation of IT project management, taking advantage of improved workflow and orchestration tools to both drive their IT projects more efficiently and to ensure consistency of process from project to project. Lifecycle Manager is designed to support these enterprise goals, both through general-purpose integration interfaces and through prebuilt integrations with key deployment tools.

For example, consider a large IT shop with tens if not hundreds of application development and maintenance projects underway at any one time. By implementing a workflow-based approach to new project initiation, this organization can help to ensure that project resources are properly allocated and the approved software development process is followed through deployment, including designated review points throughout the development lifecycle. Lifecycle Manager can be integrated into such an environment via:

- Automated Lifecycle Manager organizational group and project creation and maintenance
- Automated notifications to key architects, team leaders, and project management when project checkpoints have been reached and reviews are required
- Automated publication of Web service metadata to the enterprise runtime SOA registry when a Web service project goes into production

8.2 General-Purpose Integration Interfaces

Lifecycle Manager is designed to support easy integration with other tools through its SOAP-based integration APIs. These APIs include both inbound calls into Lifecycle Manager and outbound event notifications from Lifecycle Manager.

8.2.1 Inbound Integration to Lifecycle Manager

Lifecycle Manager exposes the following roles via its SOAP-based integration APIs:

- Asset User
- Asset Capture Engineer (ACE)
- Usage Controller
- Governance automation actions

Among other capabilities, these APIs allow enterprises to search for assets, retrieve asset details (including artifacts), initiate asset acquisition, create and edit new and existing assets, and create and

edit new and existing organizational groups and projects. This gives organizations deploying Lifecycle Manager wide-ranging flexibility to integrate their Lifecycle Manager deployment with other tools and processes.

Lifecycle Manager also provides a selected subset of its integration APIs in REST form. These APIs are focused on asset query, subscription/notification and retrieval mechanisms, and are provided to enable customers to establish lightweight Lifecycle Manager integrations using any technology capable of emitting and receiving HTTP-based messages.

8.2.2 Outbound Integration from Lifecycle Manager

By exposing its SOAP-based event engine, Lifecycle Manager makes all "events of interest" available for customers' integration and tracking purposes. Supported event types cover the full range of Lifecycle Manager capabilities, including:

- User login/logout
- Asset Searches
- Asset Usage
- Asset Acquisition
- Asset Creation and Maintenance
- Forums
- Reports
- Governance automation events
- Usage Controller activities, including:
    - Organizational Group and Project Creation and Maintenance
    - User Creation and Maintenance
    - Classification Criteria Set Creation and Maintenance
    - Asset View Creation and Maintenance
    - Profile Creation and Maintenance

Customers pick and choose from this wide range of events by creating Lifecycle Manager "listeners" to receive event messages from Lifecycle Manager. These listeners may be implemented as externally deployed Web services supporting a Lifecycle Manager-specified API (defined in WSDL) or as internal components deployed within the Lifecycle Manager application. In either case, these listeners can be implemented to process incoming events in any number of ways – integrating those events with a workflow engine, generating emails, logging information, or even calling back into Lifecycle Manager. To help customers more quickly implement their desired listeners, Lifecycle Manager provides sample listeners of both types with full source code.

In addition, Lifecycle Manager provides a preimplemented Audit Trail based on Lifecycle Manager events. This Audit Trail is configurable by any Lifecycle Manager Library Administrator, and is exposed as part of the Lifecycle Manager read-only reporting views. Audit Trail events can be subscribed to via RSS feed.

8.3 Prebuilt Lifecycle Manager Integrations

In addition to the prebuilt IDE, devops and version control repository integrations discussed in section 4, Lifecycle Manager also provides support for integrating an enterprise's development and production environments as described below. Integrations with additional run-time deployment and monitoring environments are planned for future releases.

8.3.1 Service Registry Synchronization Module

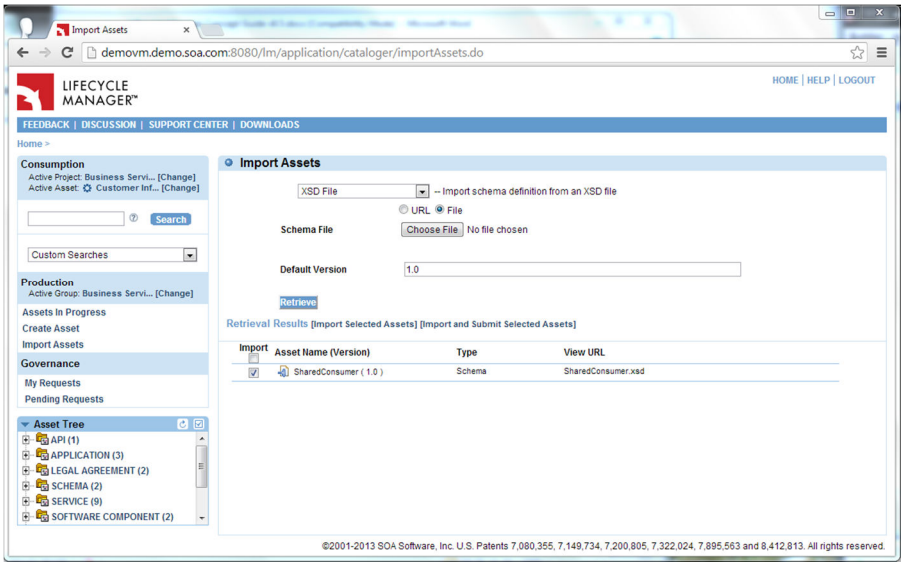
In addition to its seamless integration with Akana's Policy Manager product, Lifecycle Manager provides an integrated Service Registry Web service publication module that, in conjunction with Web service definition and deployment governance best practices, enables Lifecycle Manager to be used as a gateway to ensure only approved Web services are published to Akana Policy Manager or a UDDI-compliant registry such as TIBCO Active Matrix Registry. Such registries are used in these scenarios for operational purposes, to enable Web service clients to dynamically bind to a currently active service endpoint rather than statically implementing a fixed endpoint binding within client code, improving Web service scalability and availability within customer deployments.

The Lifecycle Manager Service Registry Synchronization Module is designed for flexibility. Because every organization's software development and deployment is different, the trigger points at which Lifecycle Manager publishes to a registry are fully configurable based on the Web service asset metadata stored within Lifecycle Manager. In addition, because of the variability in the way Web services tools produce WSDL documents, Lifecycle Manager supports two publication scenarios: a single Lifecycle Manager asset scenario, where the Lifecycle Manager asset represents both the service interface and the deployed service endpoint; and a dual Lifecycle Manager asset scenario, with one Lifecycle Manager asset representing the service interface and separate assets representing deployed instances of that service interface. In the specific case of Akana Policy Manager, Lifecycle Manager also synchronizes governance state between Lifecycle Manager and the service registry and generates runtime contracts resulting development-time acquisition requests (as discussed in Chapter 6), thereby enabling runtime better runtime access control over services, for example.

8.3.2 Semantic Web Integration

Lifecycle Manager supports both import and export of asset contents in RDF format, thereby enabling further processing and integration with Semantic Web tooling.

8.3.3 ***Asset Import Center***



Lifecycle Manager's Asset Import Center allows a Lifecycle Manager Asset Capture Engineer (ACE) to easily create new service assets within Lifecycle Manager from a list of currently ungoverned services detected in a service registry instance that has been integrated with Lifecycle Manager. For example, packaged application services that have been directly deployed to a run-time environment and their WSDLs populated into that run-time environment's service registry can be imported into Lifecycle Manager for governed use by application developers. Lifecycle Manager currently supports service imports from Akana Policy Manager and UDDI-compliant registries such as TIBCO Active Matrix Registry.

Asset Import Center also provides a quick way to create Lifecycle Manager service and schema assets from XMI, BPEL, XPDL, WSDL and XSD documents through a simple browse-and-upload interface. Automated file parsing and validation activities occur as part of the file upload and asset creation process. The XMI import process supports full configuration to map class structures represented within the XMI document to asset and relationship types within Lifecycle Manager. Both BPEL and XDPL import processes generate a root asset (e.g., Business Process) and automatically build relationships to related services specified within the imported document. These importers can optionally be configured to generate placeholder service assets for those relationships specified in the document but not yet present in Lifecycle Manager.

Spreadsheet-based imports for purposes of asset creation and/or update are also supported by Import Center. Asset import spreadsheets can be created manually or produced for a set of existing assets by exporting search results as discussed in section 5.3.3.

Finally, Import Center is designed as an extensible part of Lifecycle Manager. Custom importers can be implemented against the import framework and deployed into customer-specific installations to meet unique organizational needs.

9 How it Works: The Lifecycle Manager Architecture

9.1 Lifecycle Manager Asset Library

Akana's Lifecycle Manager Asset Library is a comprehensive index, designed to seamlessly integrate with the development environment. This library is fully internationalized, with both an English and French language version available. Other language versions may be produced based on customer demand.

9.1.1 Presentation/Client Layers

Lifecycle Manager supports a thin-client, browser-based application model to leverage its ubiquity and flexibility. Where appropriate, information is presented to the user through statically-defined HTML. However, most of the Lifecycle Manager Asset Library's sophisticated services require more dynamic interaction. Servlets and JSPs leveraging AJAX technology work with Application Components (via RMI-IIOP and XQuery) to generate the dynamic pages needed to present users with information. Client Web pages are presented to the user's browser over HTTP or HTTPS based on installation configuration settings.

Lifecycle Manager Architecture

- JEE
- Thin client
- RAD/Eclipse
- Open SOAP interfaces
- Scalable and Secure
- Relational Database Persistence
- XML Data interchange, schema and templates

9.1.2 Application Layer

The application layer of the Asset Library handles requests from the presentation layer to analyze, process, organize, and present information held at the persistence layer. Application components within this layer are built as Enterprise Java Beans (EJBs) and supporting Java objects deployed into an application server installation. The Lifecycle Manager architecture defines three major application component types:

- **Library Access:** These components manage asset and enterprise data within the Library, and provide an isolation layer that allows the asset Library to be installed in widely varying configurations. Configuration can range from a fully centralized to highly distributed with multiple, physically separate libraries interconnected and presented in a seamless and transparent manner. This sophisticated architecture also allows flexible customization of the asset structure to fit different asset types, as well as individual enterprise requirements.
- **Scoring Engine:** This component is the heart of the sophisticated search capabilities of the Asset Library. Software assets published into the Asset Library are scored based on rules established by the enterprise's Library manager. The Scoring Engine returns assets with the highest affinity, in ranked order, for presentation to the user.
- **Authentication and Authorization:** Lifecycle Manager combines LDAP integration for user authentication along with its imbedded role-based authorization model. Asset Library functions are allocated to users on a role-by-role basis. In addition, dedicated Library installations can be integrated with Single Sign On (SSO) userid and password management facilities, allowing enterprises to define a single point of management for these crucial data elements.

Standard JEE interfaces, such as JDBC and RMI-IIOP, are used to communicate between the Application Layer and other layers of the architecture.

9.1.3 Integration Layer

The Lifecycle Manager Asset Library provides a set of SOAP-based APIs for integration into the customer's environment. These APIs are WS-I Basic Profile 1.0 compliant, and thus enable cross-platform linkage between Lifecycle Manager installations and with enterprise systems of record, such as version control repositories and the Lifecycle Manager IBM Rational Application Developer / Eclipse and Microsoft Visual Studio plug-ins and add-in.

9.1.4 Persistence Layer

Akana defines four major categories of information repositories to be managed by the Asset Library:

- **Enterprise information** about users, organizational groups, profiles, etc., that define how the asset and Reference Model information within the Library is to be used.
- **Reference models** about specific industry and technical domains that allow users to rapidly identify useful business processes and associated functional capabilities that meet their current development needs.
- **Asset entries** which include the information, categorization data, artifact references, and related assets references captured in a consistent and complete form.
- **Asset Search Specifications** that describe the important functional, business, and technical characteristics of desired assets. This information is used by the Asset Library to discover the assets that best meet search criteria.

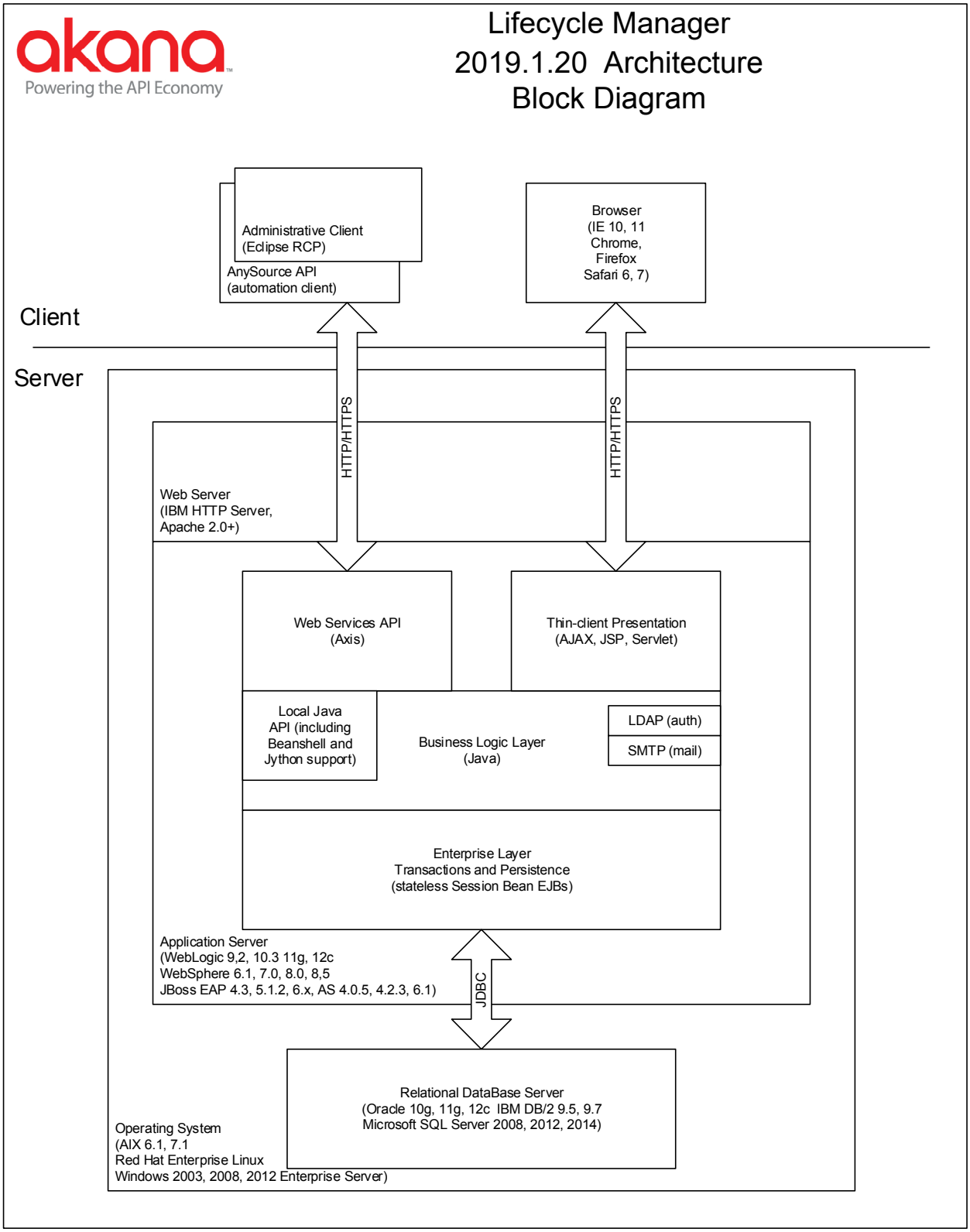
All information is managed through standard RDB and XML techniques and is exposed to report engines through a series of read-only RDB table views. Lifecycle Manager provides a set of prebuilt and automatically deployed reports against these views in Eclipse BIRT format. Other reporting engines can be easily integrated into a Lifecycle Manager deployment.

9.2 Lifecycle Manager Java IDE Plug-ins/Add-ins

Lifecycle Manager's industry-leading plug-ins for Eclipse, RAD and other Eclipse-based commercial IDEs bring the power of the Lifecycle Manager asset metadata library directly into the developers work environment. These plug-ins take full advantage of the underlying Eclipse and RAD tool framework capabilities, presenting Lifecycle Manager functions in a form that is natural and intuitive for IDE users. Akana has achieved Ready for Rational status with its RAD plug-in, a level of certification achieved by very few tool vendors.

9.3 Lifecycle Manager Add-in for Visual Studio

Lifecycle Manager's industry-leading add-in for Visual Studio brings the power of the Lifecycle Manager asset metadata library directly into the developer's work environment. This add-in takes full advantage of the underlying Visual Studio tool framework capabilities, presenting Lifecycle Manager functions in a form that is natural and intuitive for IDE users. Akana is a Microsoft VSIP Premier partner, ensuring easy access to both advanced Visual Studio APIs and Microsoft expertise to help in using those APIs effectively.


akana
Powering the API Economy
Lifecycle Manager
2019.1.20 Architecture
Block Diagram
Administrative Client
(Eclipse RCP)
AnySource API
(automation client)
Browser
(IE 10, 11
Chrome,
Firefox
Safari 6, 7)
Client
Server
HTTP/HTTPS
HTTP/HTTPS
Web Server
(IBM HTTP Server,
Apache 2.0+)
Web Services API
(Axis)
Thin-client Presentation
(AJAX, JSP, Servlet)
Local Java
API (including
Beanshell and
Jython support)
Business Logic Layer
(Java)
LDAP (auth)
SMTP (mail)
Enterprise Layer
Transactions and Persistence
(stateless Session Bean EJBs)
Application Server
(WebLogic 9,2, 10.3 11g, 12c
WebSphere 6.1, 7.0, 8.0, 8,5
JBoss EAP 4.3, 5.1.2, 6.x, AS 4.0.5, 4.2.3, 6.1)
JDBC
Relational DataBase Server
(Oracle 10g, 11g, 12c IBM DB/2 9.5, 9.7
Microsoft SQL Server 2008, 2012, 2014)
Operating System
(AIX 6.1, 7.1
Red Hat Enterprise Linux
Windows 2003, 2008, 2012 Enterprise Server)

10 Conclusion

Lifecycle Manager is an SOA design-time repository and registry that allows enterprises to represent their software development assets (SDAs) in one or more graphical Reference Models. The Lifecycle Manager SDA Library is an intelligent inventory of SDAs, their relationships to each other, and to the company's business processes and technical infrastructure.

Application developers, business analysts and architects can search this library of the company's vital software assets to identify those that best match business and technical requirements for application development and integration.

Lifecycle Manager is a very flexible asset Library which can utilize default settings for a quick start and fast return on investment. The system can be customized to fit a variety of organizational structures, business processes, and technology environments. In addition, Lifecycle Manager provides tightly-integrated plug-ins / add-in for IBM Rational Application Developer, Eclipse, and Microsoft Visual Studio, dramatically increasing developer productivity during asset search, investigation, and consumption activities.