



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

Table with 6 columns: APPLICATION NUMBER, FILING or 371(c) DATE, GRP ART UNIT, FIL FEE REC'D, ATTY. DOCKET NO, TOT CLAIMS, IND CLAIMS. Row 1: 61/165,505, 03/31/2009, 1030, 149-0215PUS

CONFIRMATION NO. 3848

FILING RECEIPT

29855
WONG, CABELLO, LUTSCH, RUTHERFORD & BRUCCULERI,
L.L.P.
20333 SH 249 6th Floor
HOUSTON, TX 77070



Date Mailed: 04/20/2009

Receipt is acknowledged of this provisional patent application. It will not be examined for patentability and will become abandoned not later than twelve months after its filing date. Any correspondence concerning the application must include the following identification information: the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. If an error is noted on this Filing Receipt, please submit a written request for a Filing Receipt Correction. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections

Applicant(s)

Thomas Louis Adrian, Dublin, CA;
Anthony George Myers, Brooklin, CANADA;

Power of Attorney:

William Hubbard--58935

If Required, Foreign Filing License Granted: 04/16/2009

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is US 61/165,505

Projected Publication Date: None, application is not eligible for pre-grant publication

Non-Publication Request: No

Early Publication Request: No

Title

Method and System for Configuration Management Database Software License Compliance

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process simplifies the filing of patent applications on the same invention in member countries, but does not result in a grant of "an international

patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at <http://www.uspto.gov/web/offices/pac/doc/general/index.html>.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, <http://www.stopfakes.gov>. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4158).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and

Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign Assets Control, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

Provisional Application for Patent Cover Sheet					
This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53(c)					
Inventor(s)					
Inventor 1					<input type="button" value="Remove"/>
Given Name	Middle Name	Family Name	City	State	Country ;
Thomas	Louis	Adrian	Dublin	CA	US
Inventor 2					<input type="button" value="Remove"/>
Given Name	Middle Name	Family Name	City	State	Country ;
Anthony	George	Myers	Brooklin	ON	CA
All Inventors Must Be Listed – Additional Inventor Information blocks may be generated within this form by selecting the Add button.					<input type="button" value="Add"/>
Title of Invention		A METHOD AND SYSTEM FOR CONFIGURATION MANAGEMENT DATABASE SOFTWARE LICENSE COMPLIANCE			
Attorney Docket Number (if applicable)		149-0215PUS			
Correspondence Address					
Direct all correspondence to (select one):					
<input checked="" type="radio"/> The address corresponding to Customer Number			<input type="radio"/> Firm or Individual Name		
Customer Number			29855		

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.	
<input checked="" type="radio"/> No.	
<input type="radio"/> Yes, the name of the U.S. Government agency and the Government contract number are:	

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

Entity Status

Applicant claims small entity status under 37 CFR 1.27

- Yes, applicant qualifies for small entity status under 37 CFR 1.27
 No

Warning

Petitioner/applicant is cautioned to avoid submitting personal information in documents filed in a patent application that may contribute to identity theft. Personal information such as social security numbers, bank account numbers, or credit card numbers (other than a check or credit card authorization form PTO-2038 submitted for payment purposes) is never required by the USPTO to support a petition or an application. If this type of personal information is included in documents submitted to the USPTO, petitioners/applicants should consider redacting such personal information from the documents before submitting them to USPTO. Petitioner/applicant is advised that the record of a patent application is available to the public after publication of the application (unless a non-publication request in compliance with 37 CFR 1.213(a) is made in the application) or issuance of a patent. Furthermore, the record from an abandoned application may also be available to the public if the application is referenced in a published application or an issued patent (see 37 CFR 1.14). Checks and credit card authorization forms PTO-2038 submitted for payment purposes are not retained in the application file and therefore are not publicly available.

Signature

Please see 37 CFR 1.4(d) for the form of the signature.

Signature	/William M. Hubbard/			Date (YYYY-MM-DD)	Mar 31, 2009
First Name	William M.	Last Name	Hubbard	Registration Number (If appropriate)	58935

This collection of information is required by 37 CFR 1.51. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 8 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. **DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. This form can only be used when in conjunction with EFS-Web. If this form is mailed to the USPTO, it may cause delays in handling the provisional application.**

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that : (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to a n other federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

PROVISIONAL PATENT APPLICATION

Title : A METHOD AND SYSTEM FOR CONFIGURATION MANAGEMENT
DATABASE SOFTWARE LICENSE COMPLIANCE

Inventors : Thomas Adrian and Anthony Myers

Docket No : 149-0215PUS Customer No : 29855

Companies have a need to understand if they are in breach of software license contracts. However, there are many complexities in a typical customer environment that may make this task difficult. Understanding what a company is licensed to use and comparing it to what is actually in use can be time consuming and/or involve many processes and technologies. There is a need for improved ways to help manage software license compliance.

Disclosed herein are several embodiments to implement a flexible license engine that may use information defined in multiple locations (*e.g.*, the CMDB, data-defined license models, contractual and certificate information) to determine software compliance for an organization.

A user may desire to view software compliance based on many variables. These variables include but are not limited to contractually (*e.g.*, is my contract in compliance), by manufacturer (*e.g.*, am I in compliance for BMC Software products), or by company being managed (*e.g.*, am I in compliance for company XYZ).

Configuration Management Database's (CMDB) are being deployed by organizations to register and control configuration items (CI's) in their infrastructure. The CMDB may capture what software is installed on what physical or virtual server as well as specific attributes (such as the number of CPU's, Operating System

manufacturer, hardware manufacturer and the number of cores on a processor) and relationships between entities. These attributes and relationships may be used to determine compliance for certain license types.

In one embodiment, an organization may define the software they are entitled to use by representing it as a software certificate. A software certificate may then be associated with a contract. The software certificate may outline the type of license model used to represent the software based on the licensing terms and conditions as defined in the associated contract.

In this embodiment, a license model defines 1) how installed software represented in the CMDB gets connected to certificates, and 2) how compliance is determined. The license model may further define a series of questions to be answered when a license certificate is created. An example question might be, "how many licenses did you purchase?" and the answer "5" might be supplied when the license certificate is created.

In one embodiment, the license model can be defined by configuring a set of rules which may allow new license models to represent complex license entitlements. In this embodiment, either no changes or simple changes to application code may be required. Also, because license models may change over time, one embodiment allows for creating license rules that are script-based and do not require application changes. Thus, allowing customers to configure the new license models via scripted rules (*i.e.*, no application code changes) for changes in license models.

The license engine may use the information in the CMDB, the certificates, and the rules in the license model to determine if a specific certificate is in or out of compliance. Furthermore, the compliance indicator may be rolled up to a contract, product, and/or manufacturer level so compliance may be presented holistically.

In another embodiment, the software license engine may also accept a scope to run against. The scope could be defined various ways. For example, it could be defined by new CI's in the CMDB, by a specific Product Name (i.e. run for all BMC Remedy Asset Management), or by a Manufacturer (i.e. BMC Software). The scope can account for the customer or future extensions to the CMDB Common Data Model (CDM) as well as federated data capabilities.

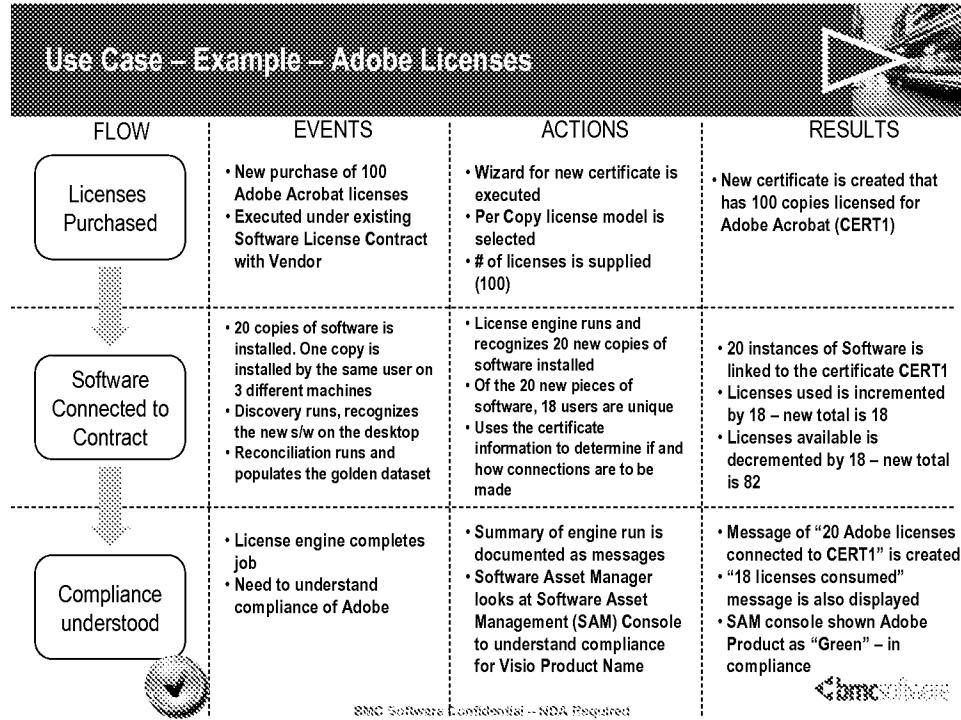
In an additional embodiment, a capability to output status and exceptions may also be part of the process. The information provided may be used to understand the results of the engine run and highlight areas that may need attention. An example of an area needing attention might be a change in compliance of a certificate from "in compliance" to "out of compliance" based on some newly discovered installed software.

Furthermore, the integration to the CMDB may allow for immediate feedback on software compliance based on changes to the infrastructure which were in turn based on reconciliation job completions. The automated updates of virtual environments or other infrastructure elements may also be leveraged to update software compliance.

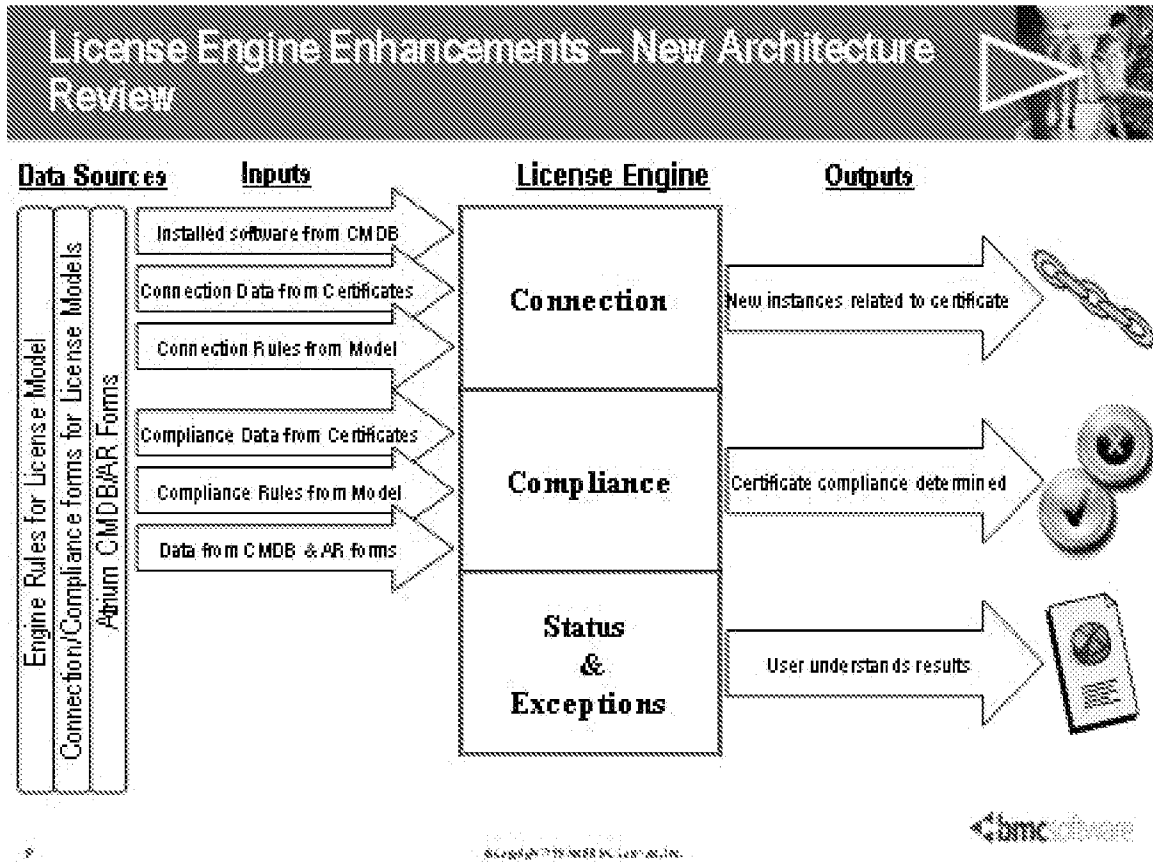
Note also, the CMDB may maintain attributes of Configuration Items and their relationships to other configuration items. These relationships make CMDB level license management attractive for different situations. For example, a software application

may be executing in a virtual machine. The license rules for this software application may dictate the rules for it are based on the number of physical processors on the machine. Because of the relationships in the CMDB, the license model may navigate back to the physical hardware to understand the number of processors.

The following diagram outlines a typical use case that illustrates one embodiment of a method and system for CMDB software license compliance.



The following diagrams illustrate the architecture and structure of different embodiments of methods and systems for CMDB software license compliance.



SWLM Basic Connection and Compliance form Model

When a new license type is registered there are two possible modes that a customer can take. One mode is the advanced mode creating a new form, and making use of the administrator tool to add fields to these forms that are specific to the license type. The other is a basic mode that makes use of a predefined form that holds a fixed set of fields that can then be mapped.

Advanced Mode

The advanced mode works as follows. When the user adds a new license type, they will be asked the mode for the questions. If they pick advanced they will be asked to provide names for a Compliance Form and a Connections form. The system will have a backend plugin which will take a template form which is provided out of the box with the core fields and workflow attached to it, and copy it to a new name based on the name that is provided by the user. The user would then make use of the AR Admin tool to add fields and workflow to the form to manage the questions and compliance information.

Basic Mode

The basic mode makes use of a pre-defined form that has a set of predefined fields on it. Those fields also have fields which will make up the labels that describe the questions related to the fields. When the user selects basic mode, they will be brought to a dialog where they can map questions to the out of the box fields.

There will also need to be work done on the certificate UI side that would read the question/field mapping form and use it to display the basic form making use of this information.

Below is an example of what this looks like when displayed in the SRM application. The labels are all data driven, and the backend form makes use of a display label and several data type fields.

Provide Information

Requester: [Edit...](#)

Name: Test Test
 Phone: ###
 Email: tom_adrian@bmc.com

Request Name: Date Required: Expected Date:

Description?

Urgency?

Impact

[Add Attachment](#) [Summary](#) [Save as Draft](#) [Submit](#)

The definition interface for this could look like this

Connection Fields

Step 1 - Enter Question

Enter Question Text:

Step 2 - Select Field

Field:

[Apply](#)

Current Mappings

Field Name	Question

[Remove](#)

[Close](#)

1.0 Solution, Product or Component Overview

Rule Engine (from now and on RLE) is standalone Remedy application with Java engine in a backend implemented as a Remedy filter plug-in. This is Rules driven engine. It provides interfaces for defining and running rules. Rules are from different types to support required from the data driven engine functionality – get data from data source, update data in data source, calculate, and loop. The architecture also allows to extend the module to support also Store that will allow to store and update date in memory before it has been written to a data source. Store is not yet supported in current release.

Current goal of the component is to provide Asset Management within Software License compliance functionality.

2.0 Architecture

2.1 Run Tag

The top of the Rules pyramid is the Run Tag. Ran Tag groups Rule Set Types, Rule Set Type groups Rule Sets, Rule Set groups Rules.

Compared to any program language Run Tag can be considered as program, Rule Set Type as service, Rule Sets as methods and Rules as low level commands.

2.2 Grouping of “Rules”

There is a need to build a structure that will allow for us to group a set of rules together in an ordered way. It would be implemented by grouping rules into Rule Sets. Each rule has a sequence of its execution order within the Rule Set.

2.3 Grouping of “Rule Sets” by “Rule Set Type”

“Rule Sets” would be grouped by “Rule Set Type”. Rule Set type also has its execution order within the Rule Set Type.

Engine allows to run all Rule Sets or some of them by passing to engine run parameters. For example

CONNECTION – will execute only Rule Sets grouped by CONNECTION type

COMPLAINCE – will execute only Rule Sets grouped by CONNECTION type

CONNECTION, COMPLIANCE – will run both

ALL – will run all rule sets for the given Run Tag

2.4 “Rule Set Type” and “Rule Set” and “Rule” - Sequence

“Rule Set Type”, “Rule Set” and “Rule” sequence define their execution order. Lower sequence plays first.

Rule Sets with lower sequence will be executed before Rule Sets with higher one. Execution order of Rules inside the Rule Set will do the same.

Sequence number value can be the same for more than one Rule Set or Rule. In that case there is no control which of them will be executed first.

2.5 Passing data to the Engine per run

Rules can not only refer to other rules results but also can refer to input parameters as where passed to the engine. This allow writing general flexible rules that will work out different data depend of parameters where passed in.

Parameters to the rules can be passed in ParametersToRules on the RLE:RuleEngineFireInterface in the following syntax:

Semicolon separated list of <Parameter Name> =<Value>;.....<Parameter Name>=<Value>;

Note. Space and “@” signs are not allowed in the parameter name!

Rules refer parameters by @parameter name@ engine will find all parameters appearances and replace it with the data as it was passed in.

Example of use:

RunTag=Live;Name=Christopher;

In this example names of the parameters to be used in the Rules are @RanTag@ and @Name@

Engine will replace @RunTag@ with Live and @Name@ with Christopher

2.6 Passing of Data and Parameter Validation

There needs to be the ability to pass data between rules. The idea is that a result would be defined as the name of the **Rule.Attribute**. Attribute could be field name as specified in the Set of Get Rule, or RESULT keyword of the Calculation Rule.

Note. Use of dot sign in the rule name is not allowed and saved only for Rule.Attribute.

Note. Rule should be represented by Rule's ID and not its name.

Note. Rule.Attribute should be surrounded by # sign this will indicate for the Engine that this is a reference to rule

2.7 More syntax stuff

In the Where statement Rule.Attribute should be surrounded by spaces.

"#RuleID.Attribute#<space>=<space>Value"

Example:

Where "#MyCRule.RESULT# = 1"

In the Set statement no spaces allowed.

"Attribute=Value"

Example:

Set "InCompliance=1"

Field name cannot contain following signs:

Two @ with no space delimiter between them

- number sign

. - dot sign

2.8 Threading Model

There are a number of issues that need to be thought about when implementing multi-threading for the engine.

The model that is proposed for the SWLM engine is the following.

There will be the ability to tag a run as a production run or simulation runs. There is only one live run data set possible, but there may be many simulation runs possible. It has been implemented by "LIVE" tag. Production Rule Sets tagged with "LIVE".

Each tagged run can only have one thread running at any given time. This is to prevent there from being conflicts with data updates that are required as part of the engine processing. If you try to run a job against the same tagged data more than once, the second run will be put into a pending form and started once the previous run has finished.

The queue will be implemented by dedicated internal form. Engine will put an entry with the run input parameters to that form each time it recognizes it has been called with the ran tag that is currently running. Each time engine finishes for some tag it will be checking if there is something pending for it to perform. If yes it will read that line, remove it from the form and start execution.

You can have multiple runs each with a different tag, however. This means one thread can be running to process a live run, which simultaneously there is a thread running processing for a simulation run.

2.9 Multiple Returns from Get Rule and Loop Rule

Get rule may get a result set that has multiple return entries. In order to iterate these results and make some action for each we need some kind of loop. For example if I have a rule set that does something like below:

Get list of Product CIs where manufacturer is = Oracle

Get a List of processors associated to the CI

Get list of cores per processor

Calculate the number of total cores

In this example I may get multiple hits from the first rule. If this is the case then for each hit I will want to process rule 2 and rule 3, and then finally rule 4 should be processed.

This has been handled by the Loop Rule. For every entry returned by get rule inside the loop scope Engine will call the next rule recursively. So with the loop rule our example above would look like following in pseudo code:

```
Loop Rule start
{
    Get Rule GetCIs (of Product CIs where manufacturer is = Oracle)
    Get Rule GetCPUs (where CPU_id = #GetCIs.id#)
    Loop Rule Start (we need this loop to be kind of escape from looping on the
    results of the following GetCores rule, as we need CalculateCores to play only
    once)
    {
        Get Rule GetCores(where Core_id = #GetCPUs.id#)
    }
    Loop Rule End
    Calculate Rule CalculateCores (using GetCores.COUNT)
}
Loop Rule end
```

For Get Rule defined within a loop, the only relevant and accessible attribute outside the loop is COUNT. If other rule despite this will try to access any other attribute an exception will occur. Run will be aborted and its status will be “Completed with errors”. In the exception table will be written a message BMC-RLE000014E “The value for [{0}] attribute does not exist.” Where [{0}] is a placeholder for the attribute it tries to access.

2.10 How to handle external data

The design point for how the engine will handle gets from external data will be via Vendor form plug-in or View forms to communicate to the external data source, or make use of the view form functionality to communicate to the database directly. Then all interactions with the external source would be the same as how we interact with regular AR forms.

3.0 Solution, Product or Component Design

The component is divided into two main parts. One is an AR deployable application the other is the backend Java engine.

AR Application.

This is stand alone deployable application which main purpose is to provide interfaces to create and manage rules and also Remedy data structure to store them. It also manages the engine’s configuration like log path, log level and so on.

Backend Java Engine.

This is the rules driven engine. It reads rules and executes them according to their sequence.

3.1 Rule Set Type

Rule Set Types defined by user and stored in the in the Rule Set Types Registry form. Rule Set Type should have meaningful name and referred later on by the Engine in run time its unique Id. Rule Sets grouped by Rule Set Type Id, this allows grouping of Rule Sets according to their logical meaning at the run time. For instance if has created two different Rule Set Types – CONNECTION and COMPLAINE and has created few Rule Sets under each, he will be able to run Engine with all the Rules under those Rule Sets by passing to the Engine command that will include these two types.

Rule Set Types has execution sequence this in order to implement *RUN ALL Rule Set Types* functionality – passing string “ALL” in the Rule Set Type and “Your Company” in the Company fields as parameters to the Engine for run will fire all rules of the given Company (of the given Run Tag).

3.2 Rule Set

Rule only exists in the scope of the Rule Set. Each Rule Set has its run sequence – lower sequence fires before the upper one. This gives to user an ability to set the order of rules execution. Rule Set type defined by the user. The first parameter passed is column separated list of types which defines which type/s of the rule set need to be run. ALL states for all.

3.3 Connections

Rule Set Types and Rule Sets are only relevant in the context of their container. Rule Set's container is Rule Set Type and Rule Set Type's one is Run Tag.

To support reusability of those entities information as Sequence and Status are stored in the RLE:RBA:Tag2Type_Type2Set_Relationship table which is relational table between Rule Set Type to Run Tag and Rule Set to Rule Set Type.

This way same Rule Set can be used by different Rule Set Types, for each sequence and status can be different.

3.4 AR Application – backend engine relationships

On submitting record to the RLE:FireEngineInterface a workflow triggered. It creates a record with the parameters passed in to the run on the RLE:RunHistory with status pending. It then submit the run to the backend via filter api. Java engine checks whether it already runs another thread for the same Run Tag. If yes it returns and leaves the run in the table in the same state – Pending. If not it changes it immediately to Running and process it. When the run completed Engine changes its status to Completed, which trigger to another workflow to start, find the oldest pending run and pass it to the Engine.

There is a problem with this solution as if the run take much time end user that has submitted a run via Remedy User Tool gets not responding screen until the time out occurs. As a solution for this problem an escalation can be used. On submitting a run to the RLE:FireEngineInterface only a record on the RLE:RunHistory will be created in Pending status. An escalation that will play every 1 minute will submit all pending runs to the backend engine. This way user will get control right away after submitting a run and will required to check the status later on the RunHistory form. Possible problem with this solution is that if escalation runs in a single thread – meaning if there are few runs to submit it submits first, waits for the result and then continue to the next. It is still not that bad as it in any case will not wait more than a time out.

Another point of interaction is the filter API plug-in starting. At this point something has to abort all runs if any that are in Running status. On initialization Engine changes EngineInReset field on the RLE:Configuration form which cause to workflow to run. It changes status of all runs with status Running to Aborted.

3.5 Types of Rules

3.5.1 Get Rule – Rule that defines the scope and source of data to fetch and fields to fetch

- Name of the Rule
- Rule Set ID
- Rule ID
- Sequence
- Type (AR Form or CMDB class or Storage)
- Select (Specifies names of the fields to get)
- From (Name of AR Form or CMDB class or Storage)
- Where (Parameterized Qualification)

Fields that retrieved will be stored in memory using the notation RuleName.FieldName – field name as specified in the Select. These fields can be later use by other rules in the same Rule Set by #RuleID.FieldName#.

Get Rule by default includes COUNT function which returns count of found entries. If there is a need just to count a number of entries that meet the Where qualification it is possible to leave Select field empty then GetRule will not contain any fields but <GetRule ID>.COUNT will still return a number of found.

3.5.2 Complex Query Rule

Complex rule implement the same functionality as the Get Rule – retrieve data from data source, but it different in two aspects. First its only data source is CMDB. Second unlike Get Rule that contains all the information needed for query in one record Complex Rule needs a serial of records to define it's query. It calls QueryByPath CMDB api to issue a query.

- RuleID – Rule ID. There are will be more than one record with the same RuleID as now primary key is RuleID and Serial Number
- Serial Number – Position in the rule
- Sequence – Rule's sequence in the rule set
- Class – CMDB class
- Type – Type of the object Relationship or Regular. Engine will create class accordingly
- RelationshipDirection – NULL / CMDB_RELATIONSHIP_DIRECTION_OUT, CMDB_RELATIONSHIP_DIRECTION_IN, more?
- Select – Fields to select
- Query – Query statement / NULL
- NestedQuery – True/False
- SelectorForNested Attribute – Attribute of the current class specifies join key with the nested query

For detailed diagram see ComplexQueryRule.vsd at

<http://sharepoint.bmc.com/RnD/ProjectCenter/aristotle/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2fRnD%2fProjectCenter%2faristotle%2fShared%20Documents%2fSDD%27s%2fAsset%2fEngine&SortField=DocIcon&SortDir=Asc&View=%7b674D05D3%2d662B%2d4612%2dA4EC%2d7B927B82C32C%7d>

3.5.3 Calculation Rule – Arithmetical calculation

- Name of the rule
- Rule Set ID
- Rule ID
- Sequence
- Expression
(#Deployed + #Found for example).
Calculate Rule consolidates comparison functionality as well.
Following operators supported: Operators: +, -, /, *, ^ Boolean Operators: <, >, =, &, |, !, <>, >=, <=
Paranthesis: (, {, [

For current release this rule will support only Integers!

Result of the Calculation Rule written into the memory (i.e. RuleName.RESULT) and later be used by other rules in the same Rule Set.

3.5.4 Update Rule – Updates target data source with the results of previous rules in the Rule

Set

- Name of Rule
- Rule Set ID
- Rule ID
- Sequence
- Target Type (Use Enum for 1/2/3 for AR Form or CMDB class or Storage)
- Target (Name of AR Form or CMDB Class or Storage)
- Set – a list of semicolon separated pairs of field/s and value/s in the following structure:
field=value;field=value;...
No spaces allowed!
- Where (Parameterized Qualification). If empty then Engine will treat update rule as insert, and will create a new entry. Otherwise it will try to update a record. Meaning it will first try to receive an entry from the server and only then to update it.
Always true like "1=1" means that it matches all the records in the target table. It will behave then as specified by "If any requests match" parameter.
- If any requests match
"If any requests match" resolves a situation where more than one entry found according to Where clause.

It has 3 optional values – 1,2,3 staying for:

1 - Modify all matching records.

2 - Do not update, issue a warning and continue. Run status in this option will be “Completed with warnings”

3 - Do not update, issue an error and abort run. Run status in this option will be “Completed with errors”

3.5.5 Storage Rule – creates a storage in the memory – not implemented in current version

- Name of rule
- Rule Set ID
- Rule ID
- Sequence
- Attributes – a list of semicolon separated pairs of: Attribute name, Attribute type (Char and Int).

3.5.6 Loop Rule

- Name of rule
- Rule Set ID
- Rule ID
- Sequence
- Operation – 1 (LOOP START) or 2 (LOOP END)

Currently there is no validation mechanism that will check if all loops are closed

3.6 Interfaces

3.6.1 RLE:RuleEngineFireInterface

FieldID	Field Name	Type	Description
Core fields TBD			
	Rule Set Type	Char	Input. Semicolon separated list of Rule Set Type IDs. ALL will specify that all rule sets to be run.
	Company	Char	Company Name. Will be relevant and used by the Engine only if RuleSetType=ALL as it will search for all Rule Sets for that Company and run them
	ParametersToRules	Char	Input. Semicolon separated list of <Parameter Name>=<Parameter's Value>;..... Rules will use those parameters in the following way: @ParameterName@ Note, Use of dot sign in the attribute name is not allowed! Example of use: RunTag=Live;Name=Christopher; In this example names of the parameters to be used in the Rules are @RanTag@ and @Name@
	Status	Char	Output. Status

3.6.2 RLE:PassToEngineInterface – Display only form

Internal form used to pass data between the AR application to the backend java

FieldID	Field Name	Type	Description
	Run Tag ID	Char	Input. Run Tag ID to be run
	Rule Set Type	Char	Input. Semicolon separated list of Rule Set Type IDs. ALL will specify that all rule sets to be run.
	ParametersToRules	Char	Input. Parameters for run
	Run ID	Char	Input. Run ID to run

3.6.1 RLE:RunTag

FieldID	Field Name	Type	Description
Core fields TBD			
	Action	Int	Input. Possible values: 1 (CREATE), 2 (DELETE)
	Run Tag	Char	Input. Run tag.
	Company	Char	Input. Company name
	Status	Int	Output. Possible values: 0 (Success), 1(Fail)

3.6.2 RLE:RuleSet Interface

FieldID	Field Name	Type	Description
Core fields TBD			
	Action	Int	Input. Possible values: 1 (CREATE), 2 (ENABLE), 3 (DELETE) 4 (UPDATE), 5 (DISABLE), 6 (COPY)
	RuleSetName	Char	Rule Set Name
	Rule Set Type Id	Char	Input. Possible values: any string the set will be grouped by. ALL would be restricted and if used an exception will be generated.
	Rule Set ID	Int	Output/Input. Rule Set ID.
	Status	Int	Output. Possible values: 0 (Success), 1(Fail)

3.6.1 RLE:Connections Interface – Display only form

Defines connection between Ran Tag to Rule Set Type and Rule Set Type to Rule Set

FieldID	Field Name	Type	Description
	Action	Int	Input. Possible values: 1 (CREATE), 2 (ENABLE), 3 (DELETE) 4 (UPDATE), 5 (DISABLE), 6 (COPY)
	Left ID	Int	Input Left ID of the connection Run Tag ID or Rule Set Type ID depends on Type
	Right ID	Int	Right ID of the connection Rule Set Type or Rule Set ID depends on Type
	Sequence	Int	Input. Rule Set Sequence.
	Type	Int	Input. 1- Specifies Run Tag to Rule Set Type connection 2- Specifies Rule Set Type to Rule Set connection

3.6.2 RLE:RuleSetType Interface

FieldID	Field Name	Type	Description
Core fields TBD			
	Action	Int	Input. Possible values: 1 (CREATE), 2 (DELETE)
	Rule Set Type	Char	Input. Possible values: any string except ALL that would be restricted and if used an exception will be generated.
	Run Tag Id	Char	Input. Run Tag to be tied to.
	Sequence	Int	Sequence.
	Status	Int	Output. Possible values: 0 (Success), 1(Fail), 2(Duplicate)

3.6.3 RLE:RuleSetActionInterface

FieldID	Field Name	Type	Description
Core fields TBD			
	Action	Int	Input. Possible values: 1 (ENABLE), 2 (DISABLE)
	Rule Set Type	Char	Input. Possible values: any string except ALL that would be restricted and if used an exception will be generated.

FieldID	Field Name	Type	Description
	Status	Int	Output. Possible values: 0 (Success), 1(Fail)

3.6.4 RLE:LoopRule Interface

FieldID	Field Name	Type	Description
Core fields TBD			
	Action	Int	Input. Possible values: 1 (CREATE) 2 (UPDATE), 3 (DELETE)
	Rule Set ID	Char (GUID)	Input. Rule Set ID
	Rule ID	Char	Output.
	Sequence	Int	Input. Sequence of the rule. This should be output on create.
	Operation	Int	1 (LOOP START), 2 (LOOP END)
	Status	Int	Output. Possible values: 0 (Success), 1(Fail)

3.6.5 RLE:GetRule Interface

FieldID	Field Name	Type	Description
Core fields TBD			
	Action	Int	Input. Possible values: 1 (CREATE) 2 (UPDATE), 3 (DELETE)
	Name	Char	Input. Rule name
	Rule Set ID	Char (GUID)	Input. Rule Set ID
	Rule ID	Char (GUID)	Output.
	Sequence	Int	Input. Rule sequence.
	Source Type	Int	Input:1 (CMDB), 2 (AR), 3 (STORAGE)
	From	Char	Input. Data source schema name. In case of CMDB syntax should be <Namespace>:<Class name>
	Select	Char	Input. Comma separated list of fields to be retrieved with each entry.
	Where	Char	Input. Where clause as remedy qualification. If referring to the results of previous rules then use #RuleId.Attribute# If referring to input parameters the use @Parameter name@
	Status	Int	Output. Possible values: 0 (Success), 1(Fail)

3.6.6 RLE:CalculationRule Interface

FieldID	Field Name	Type	Description
Core fields TBD			
	Action	Int	Input. Possible values: 1 (CREATE) 2 (UPDATE), 3 (DELETE)
	Name	Char	Input. Rule name.
	Rule Set ID	Char (GUID)	Input. Rule Set ID
	Rule ID	Char (GUID)	Output.
	Sequence	Int	Input. Rule sequence
	Expression	Char	Input. Any valid compare expression (Like expression > expression) Operators: +, -, /, *, ^ Boolean Operators: <, >, =, &, , !, <>, >=, <= Paranthesis: (, {, [
	Status	Int	Output. Possible values: 0 (Success), 1(Fail)

3.6.7 RLE:UpdateRule Interface

FieldID	Field Name	Type	Description
Core fields TBD			
	Action	Int	Input. Possible values: 1 (CREATE) 2 (UPDATE), 3 (DELETE)
	Name	Char	Input. Rule name
	Rule Set ID	Char (GUID)	Input. Rule Set ID
	Rule ID	Char (GUID)	Output.
	Sequence	Int	Input. Rule sequence
	Target Type	Int	Input:1 (CMDB), 2 (AR), 3 (STORAGE)
	Target	Char	Input. Name of the AR Form or CMDB class or Storage
	Set	Char	Input. Set command – a list of semicolon separated pairs of field/s and value/s in the following structure: field=value;field=structure; No spaces allowed!
	Where	Char	Input. The where clause as remedy qualification. If referring to the results of previous rules then use #RuleId.Attribute# If referring to input parameters the use @Parameter name@
	If Any Request Match	Selection	Possible values: 1. Modify all matching records. 2. Do not update, issue a warning and continue. Run status in this option will be "Completed with warnings" 3. Do not update, issue an error and abort run. Run status in this option will be "Completed with errors"
	Status	Int	Output. Possible values: 0 (Success), 1(Fail)

3.6.8 RLE:StorageRule Interface

FieldID	Field Name	Type	Description
Core fields			
	Action	Int	Input. Possible values: 1 (CREATE) 2 (UPDATE), 3 (DELETE)
	Name	Char	Input. Rule name
	Rule Set ID	Char (GUID)	Input. Rule Set ID
	Rule ID	Char (GUID)	Output.
	Sequence	Int	Input. Rule sequence
	Attributes	Char	Input. Semicolon separated list of pairs: Attribute name, Attribute type
	Status	Int	Output. Possible values: 0 (Success), 1(Fail)

3.6.9 RLE:ComplexRuleInterface

FieldID	Field Name	Type	Description
Core fields			
	Action	Int	Input. Possible values: 1 (CREATE) 2 (UPDATE), 3 (DELETE)
	Name	Char	Input. Rule name
	Rule Set ID	Char (GUID)	Input. Rule Set ID
	Rule ID	Char (GUID)	Input/Output.
	Sequence	Int	Input. Rule sequence

3.6.10 RLE:ComplexRuleQueryInterface

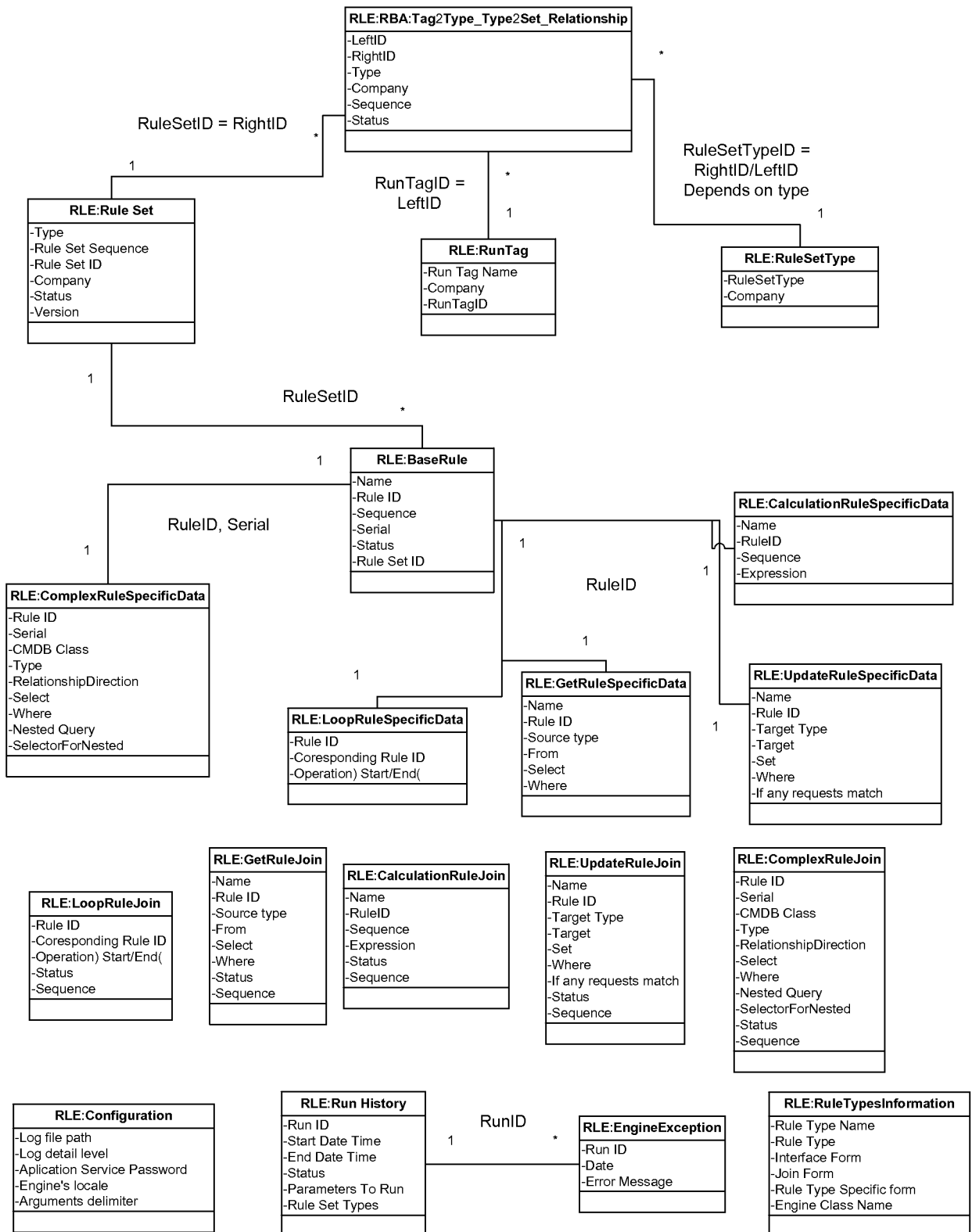
FieldID	Field Name	Type	Description
Core fields			
	Action	Int	Input. Required. Possible values: 1 (CREATE) 2 (UPDATE), 3 (DELETE)
	Rule ID	Char (GUID)	Input. Required.

FieldID	Field Name	Type	Description
	Sequence	Int	Input. Required. Qualification sequence
	Class Name	Char	Input. Required. Class name
	Name Space	Char	Input. Required. Name Space of the class
	Type	Selection	Input. Required.Type of the object Relationship or Regular. Engine will create class accordingly Optional values: RELATIONSHIP REGULAR
	RelationshipDirection	Selection	Input. Optional. Optional values: CMDB_RELATIONSHIP_DIRECTION_OUT CMDB_RELATIONSHIP_DIRECTION_IN Default value is CMDB_RELATIONSHIP_DIRECTION_OUT
	Select	Char	Input. Optional. Comma separated list of fields to be retrieved with each entry. Might be null if no fields need to be selected.
	Where	Char	Input. Optional. Where clause as remedy qualification. If referring to the results of previous rules then use #RuleId.Attribute# If referring to input parameters the use @Parameter name@
	HasNestedQuery	Selection	Input. Optional. Optional values: TRUE FALSE
	NestedQueryOperator	Selection	Input. Optional. Optional values: IN NOT_IN
	SelectorForNestedAttribute	Char	SelectorForNestedAttribute – Attribute of the current class specifies join key with the nested query.
	FirstElement	Selection	Input. Optional. Optional values: TRUE - specifies first element of nested query. FALSE

3.6.11 RLE:RuleActionInterface

Enables, Disables or Delete existing rule

FieldID	Field Name	Type	Description
Core fields			
	Action	Int	Input. Possible values: ENABLE, DISABLE, DELETE
	Rule ID	Char	Input.
	Status	Int	Output. Possible values: 0 (Success), 1(Fail)



3.6.12 RLE:RunTag – Regular form

FieldID	Field Name	Type	Description
Core fields			
	Run Tag	Char	Run Tag name.
	Run Tag Id	Char	Run Tag Id
	Company	Char	Company
	Version	Int	Version

3.6.1 RLE:RuleSetType – Regular form

FieldID	Field Name	Type	Description
Core fields			
	Type Name	Char	Rule Set Type Name
	Type Id	Char	Rule Set Type Id
	Status	Selection	Enabled, Disabled, Marked for Delete
	Version	Int	Version

3.6.2 RLE:RuleSet – Regular form

It can be more than one Rule Set with the same name, but not for the same company. This will be forced on the filter level and not database level. Rule Set ID (ID) is unique index.

FieldID	Field Name	Type	Description
Core fields			
	Name	Char	Rule Set Name
	Type	Char	Rule Set Name
	Id	Char	Rule Set ID. (Unique index)
	Status	Selection	Enabled, Disabled, Marked for Delete
	Version	Int	Version

3.6.3 RLE:BaseRule

On creating of each rule two entries has been created on two form. One is on the RLE:BaseRule that store basic information of the rule, like status and sequence and the other on the RLE:<RULE TYPE>SpecificData form that stores type specific data. This is to enable global workflows for all rules. The foreign key that defines relationship between base and specific is Rule ID on both.

FieldID	Field Name	Type	Description
	Rule ID	Char	Rule ID
	Name	Char	Rule Name
	Sequence	Int	Sequence of the rule.
	Serial	Int	Relevant only for Complex Rule, 0 for the rest. Complex rule can be defined as a number of consequent rules. Serial defines the order of these sub rules within the main rule.
	Status	Selection	Enabled, Disabled, Marked for Delete

3.6.4 RLE:LoopRuleSpecificData – Regular form

FieldID	Field Name	Type	Description
Core fields			
	Rule Set ID	Char	Rule Set ID
	Rule ID	Char	Rule ID
	Corresponding Rule ID	Char	Id of the corresponding loop rule – of the loop start for the loop end and visa versa
	Operation	Int	1 (LOOP START), 2 (LOOP END)

3.6.5 RLE:GetRuleSpecificData – Regular form

FieldID	Field Name	Type	Description
Core fields			
	Rule Set ID	Char	Rule Set ID
	Rule ID	Char	Rule ID
	Source Type	Int	1 (CMDB), 2 (AR), 3 (STORAGE)
	From	Char	Name of the AR Form or CMDB class or Storage
	Select	Char	Comma separated list of fields to be retrieved with each entry.
	Where	Char	Where clause.

3.6.6 RLE:ComplexRuleSpecificData

FieldID	Field Name	Type	Description
Core fields			
	Rule Set ID	Char	Rule Set ID
	Rule ID	Char	Rule ID
	Serial	Int	Sub sequence in the rule
	CMDB Class	Int	Name of the CMDB Class
	Type	Selection	Type of the object 1 for Relationship 2 for Regular
	RelationshipDirection	Char	NULL / CMDB_RELATIONSHIP_DIRECTION_OUT, CMDB_RELATIONSHIP_DIRECTION_IN, more?
	Select	Char	Comma separated list of fields to be retrieved with each entry.
	Where	Char	Where clause.
	NestedQuery	Selection	1 – True, 2 - False
	SelectorForNested	Char	Attribute of the current class specifies join key with the nested query

3.6.7 RLE:CalculationRuleSpecificData– Regular form

FieldID	Field Name	Type	Description
Core fields			
	Rule Set ID	Int	Rule Set ID
	Rule ID	Int	Rule ID
	Expression	Char	Any valid compare expression (Like expression > expression)

3.6.8 RLE:UpdateRuleRuleSpecificData – Regular form

FieldID	Field Name	Type	Description
Core fields			
	Rule Set ID	Int	Rule Set ID
	Rule ID	Int	Rule ID
	Target Type	Int	1 (CMDB), 2 (AR), 3 (STORAGE)
	Target	Char	Name of the AR Form or CMDB class or Storage
	Set	Char	Set command
	Where	Char	The where clause.
	If Any Request Match	Selection	Possible values: 1. Modify all matching records. 2. Do not update, issue a warning and continue. Run status in this option will be "Completed with warnings" 3. Do not update, issue an error and abort run. Run status in this option will be "Completed with errors"

3.6.9 RLE:StorageRuleRuleSpecificData – Regular form

FieldID	Field Name	Type	Description
Core fields			
	Rule Set ID	Int	Rule Set ID
	Rule ID	Int	Rule ID
	Attributes	Char	Semicolon separated list of doubles: Attribute name, Attribute type

3.6.10 RLE:Configuration– Regular form

FieldID	Field Name	Type	Description
Core fields			
	Log file path	Char	The path to the log file of the Engine
	Log detail level	Char	Date indicates the date of start and end of the run
	Application Service Password	Char	Encrypted password for application service internal user for remedy
	Arguments delimiter	Char	Specifies special character to delimit between parameters in the “parameters to run” and fields to be set in the Set statement of the Update Rule. Default is “;”
	Engine in reset	Char	Hidden field. Engine Switches its value between True and False on startup. This trigger to a workflow to start. This workflow searches the RLE:RunHistory form for all runs with status Running and change it to aborted.
	Engine’s Local	Char	Engine’s locale. Default is EN for English

3.6.11 RLE:Run History

FieldID	Field Name	Type	Description
Core fields			
	Run ID	Char	Run ID
	Run tag	Char	Run Tag Id
	Company	Char	Company field
	ParametersTo Rules	Char	Parameters to rules
	Start Date Time	Char	Start date and time of the run
	End Date Time	Char	End date and time of the run
	Status	Int	Status of the Run - Pending, Started, Running, Completed, Completed with Errors, Aborted

3.6.12 RLE:EngineExceptions

FieldID	Field Name	Type	Description
Core fields			
	Run ID	Char	Run ID
	Date Time	Char	Date time of the error
	Error Message	Char	Detailed Message

3.6.13 RLE:RuleTypesInformation

Internal form, that used by the interfaces on rules creation. It also might be used in the future by the backend in the future to describe new rule type. Four entries (for Calculation, Get, Loop, Storage, Update rules) on this forms are installed within the application on the target AR System.

FieldID	Field Name	Type	Description
Core fields			
	Rule Type Name	Char	Name of the Rule Type
	Rule Type	Int	Possible values: 1 – Calculation, 2 - Get, 3 –Loop, 4 – Storage, 5 – Update
	Interface Form	Char	Name of the interface form for rule creation for example RLE:CalculationRuleInterface

FieldID	Field Name	Type	Description
	Join Form	Char	Name of the join form between Base and Specific, for example RLE:CalculationRuleJoin
	Rule Type Specific form	Char	Name of the specific form. For example RLE:CalculationRuleSpecificData
	Engine Class Name	Char	Not in use in current version

3.6.14 Vendor Form to display Engine log, History and so - is not implemented in current version

3.6.15 RLE:EngineConsole– Display only form – is not implemented in current version

This form should be in format of the administrator console like the one for the reconciliation

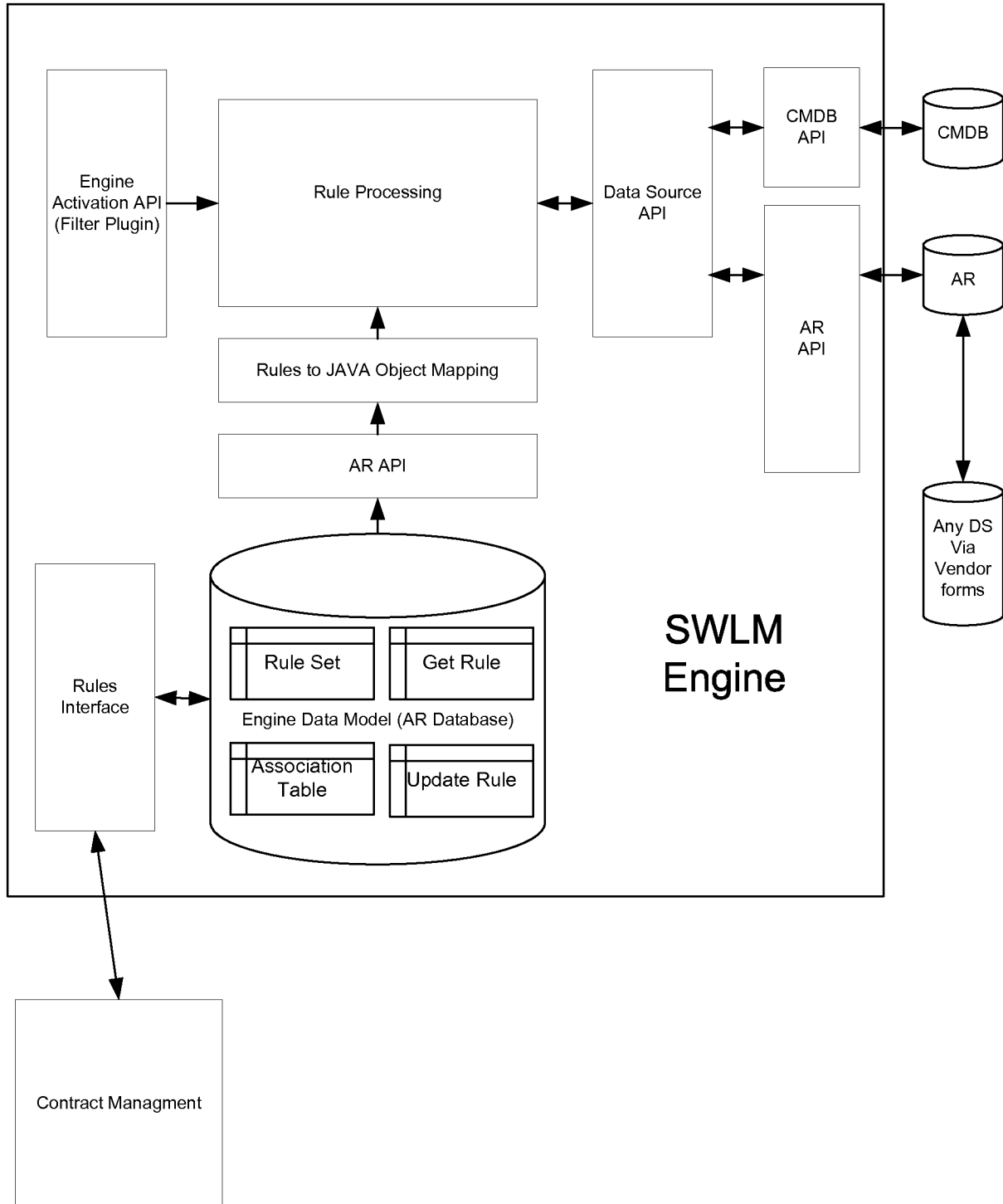
Form name	Form code	Type	Description	Registered	Implemented
RLE:CalculationRule	REA	Regular	Calculation rule	Yes	
RLE:CalculationRule Interface	REB	Regular	Calculation rule interface	Yes	
RLE:GetRule	REC	Regular	Get Rule	Yes	
RLE:GetRule Interface	RED	Regular	Get Rule interface	Yes	
RLE:LoopRule	REE	Regular	Loop Rule	Yes	
RLE:LoopRule Interface	REF	Regular	Loop Rule interface	Yes	
RLE:RuleSet	REH	Regular	Rule Set	Yes	
RLE:RuleSet Interface	REJ	Regular	Rule Set interface	Yes	
RLE:RuleSetTypeRegistry	REI	Regular	Stores registry of Rule Set Type	Yes	
RLE:RuleSetTypeRegistry Interface	REK	Regular	Rule Set Type Registry interface	Yes	
RLE:RunHistory	REM	Regular	Stores run history	Yes	
RLE:StorageRule	REN	Regular	Storage Rule	Yes	
RLE:StorageRule Interface	REO	Regular	Storage Rule interface	Yes	
RLE:UpdateRule	REP	Regular	Update Rule	Yes	
RLE:UpdateRule Interface	RER	Regular	Update Rule interface	Yes	
RLE:RuleEngineFireInterface	RES	Regular	Rule Engine application activation api.	Yes	
RLE:Configuration	RET	Regular	Engine configuration form	Yes	
RLE:PassToEngineInterface		Display Only	Submit here an entry will cause a filter to fire and pass data to the filter api of the Java Engine	No	
RLE:RunTagRegistry	REU	Regular	RunTagRegistry	Yes	
RLE:					
RLE:EngineExceptions	REW		Stores Exceptions	Yes	
RLE:Tag2Type_Type2Set_Relationship	RBA		Many to many connections between RunTag to/from RuleSetType and RuleSetType to/from RuleSet	Yes	
RLE:ComplexRuleInterface	RBB	Regular	Complex Rule header interface	Yes	
RLE:ComplexRuleQueryInterface	RBC	Regular	Complex Rule single class qualification interface	Yes	

3.7 Error Messages

Interfaces returns to the caller following error messages.

ID	Text	Description	Active Link/Filter
45546	Rule Set doesn't exist	The caller is trying to modify Rule Set or trying to create a Rule within a Rule Set, but passing incorrect Rule Set Id	
45547	Rule doesn't exist	The caller is trying to update Rule, but passing incorrect Rule Id	
45548	Duplicate Name	Thrown on creation of Rule Set or Rule if specified name already exists.	
45549	Rule Set Type doesn't exist	Thrown on Rule Set creation if Rule Set Type Id specified is incorrect.	
45550	Unknown error	Unknown error occurred on Rule Set or Rule creation.	
45551	Missing Start Loop Rule ID	On Loop Rule creation of End Operation Rule Id of the corresponding Start Rule wasn't specified.	
45552	ID <id> is already in use.	On creating rule/rule set/rule set type/run tag passed in id is already in use. Regenerate and call again.	

3.8 Engine modules



3.8.1 Engine Activation API (Filter Plugin)

This module defines AR API interface to fire engine. It defines input parameters and the way caller passes it. Requirement is not to allow parallel runs with the same Company and Run Tag to avoid conflicts. We can resolve it by managing one main thread in the Engine which will recognize whether to proceed or push to a waiting queue some Runs. This approach will require thread managing. We prefer to take advantage of the thread managing by

AR. The resolution will be a queue that will be managed by the Rules Engine AR application. It will decide if execute Run or push it to a waiting queue. Java Engine will have to signal upon completion of each run. See "Runs waiting queue and Rule Engine run API.vsd"
[http://sharepoint.bmc.com/RnD/ProjectCenter/aristotle/Shared%20Documents/Software%20License%20Mgmt%20\(Asset\)/Techninca%20Documentation/Engine%20Documentation/Runs%20waiting%20queue%20and%20Rule%20Engine%20run%20API.vsd](http://sharepoint.bmc.com/RnD/ProjectCenter/aristotle/Shared%20Documents/Software%20License%20Mgmt%20(Asset)/Techninca%20Documentation/Engine%20Documentation/Runs%20waiting%20queue%20and%20Rule%20Engine%20run%20API.vsd)

3.8.2 Rules Interface

AR API interface to manage rules.

3.8.3 Rules to Java object module

Java representation of the rules. For each rule that Engine read a java object been created that represent the rule's attributes like FROM, SEQUENCE, WHERE etc in the memory.

3.8.4 Data Source API

Common layer to deal with any DS data. Engine Rule Processor doesn't need to know about differences between CMDB, AR or Storage and this layer resolves this. For example to execute Get command engine will call Get command on this layer and get back results in unified format no matter what is the DS.

3.8.5 Rule Processor.

This is the heart of the engine. Here the rule sets and rules inside rule sets executed in the right order.

3.9 Exception handling model

All Java or AR exceptions that were thrown during a work of the Rule Engine will be wrapped in `com.bmc.itsm.rle.exceptions.RLEException`. All `com.bmc.itsm.rle.exceptions.RLEException` will be logged with ERROR level of Log4j tool in the AR Plugin Server log file. Besides the exception message will be put in the "RLE:EngineExceptions" AR form.

In case of exception was thrown during an AR Server timeout, user application will get same message of exception that logged.

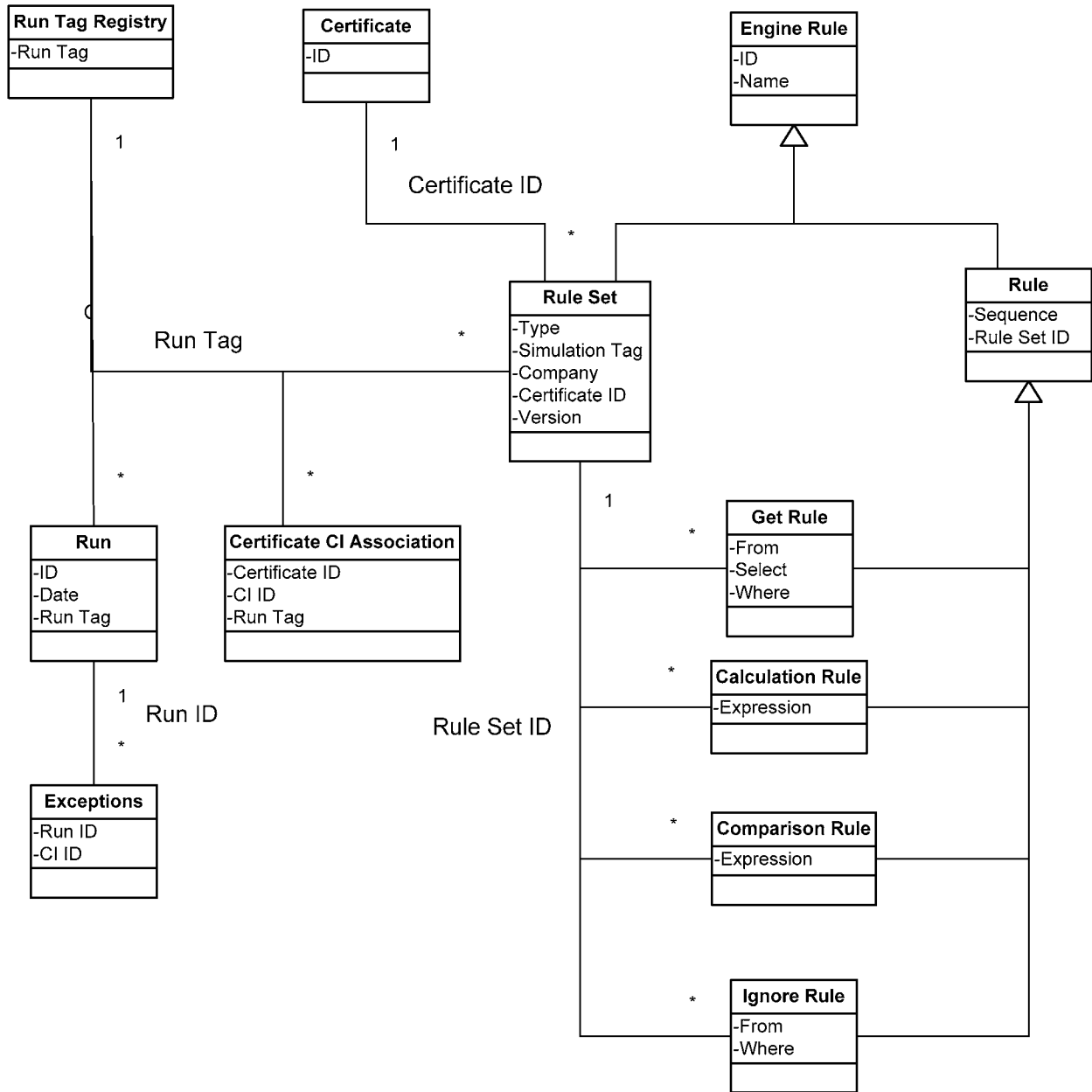
The exception messages should be kept as list of properties in the separate text file. This solution guarantees possibility to change any message (according to a locale as sample) without compilation of java source.

3.10 Logging model

The Rule Engine uses Log4j tool for logging messages and java exceptions in the AR Plugin Server log file. This tool works according to specified log level (as INFO, DEBUG, ERROR, WARN and etc.). During the initializing process the Rule Engine get this level from "RLE:Configuration" AR form. In case changing log level, the Rule Engine should be restarted.

3.11 Engine classes module

See following diagram



4.0 Use Cases

4.1 Processing of the Connection Rules

Actually Engine doesn't know about connection rules, but it provides that functionality in the scope of the SWLM. Input parameters - there is the ability to pass parameters to the engine. It can be used as scope qualification for rules.

Get Rule - would run a query against the CMDB to get the list of CIs that match the scope that is being looked at. Update Rule - for all the CIs that are found to be tied to a particular certificate, the list of all CIs will be updated with the certificate ID.

Update Rule - any CI that does not have a certificate id associated to it will then be fed to the exception processing model.

Update Rule - finally there will be a rulset to commit the relationships to the database. This will walk the list of CIs in memory, and push the data to the relationships table.

5.0 Engine processing algorithm

See

[http://sharepoint.bmc.com/RnD/ProjectCenter/aristotle/Shared%20Documents/Software%20License%20Mgmt%200\(Asset\)/Techninca%20Documentation/Engine%20Documentation/Rule%20Engine%20algorithm.doc](http://sharepoint.bmc.com/RnD/ProjectCenter/aristotle/Shared%20Documents/Software%20License%20Mgmt%200(Asset)/Techninca%20Documentation/Engine%20Documentation/Rule%20Engine%20algorithm.doc)

1.0 Solution, Product or Component Overview

Exceptions in Software License Management (SWLM) consist of Connection exceptions and Compliance exceptions, and occur after a license job has been run.

Connection exceptions occur when the rules engine is unable to connect a CI to a License Certificate or when the rules engine determines that a CI can potentially be connected to multiple License Certificates.

Compliance exceptions occur when the rules engine connects a CI to a License Certificate and that certificate has more CIs than it has licenses for.

This feature consists of capturing these exceptions from a job run, summarizing the information, creating Inbox entries (if the system is configured to do so) and displaying the job run results (exceptions + successful connections) to the user.

2.0 Architecture

2.1 Process Flow

License Jobs are run from the “Manage License Job” console. Once a run is kicked off, the rules engine determines the scope of the run (i.e. determines which CIs to process) and then processes these CIs. As the rules engine processes the CIs, the data will be examined to determine if it is an exception, and stored in an exception table if that is the case.

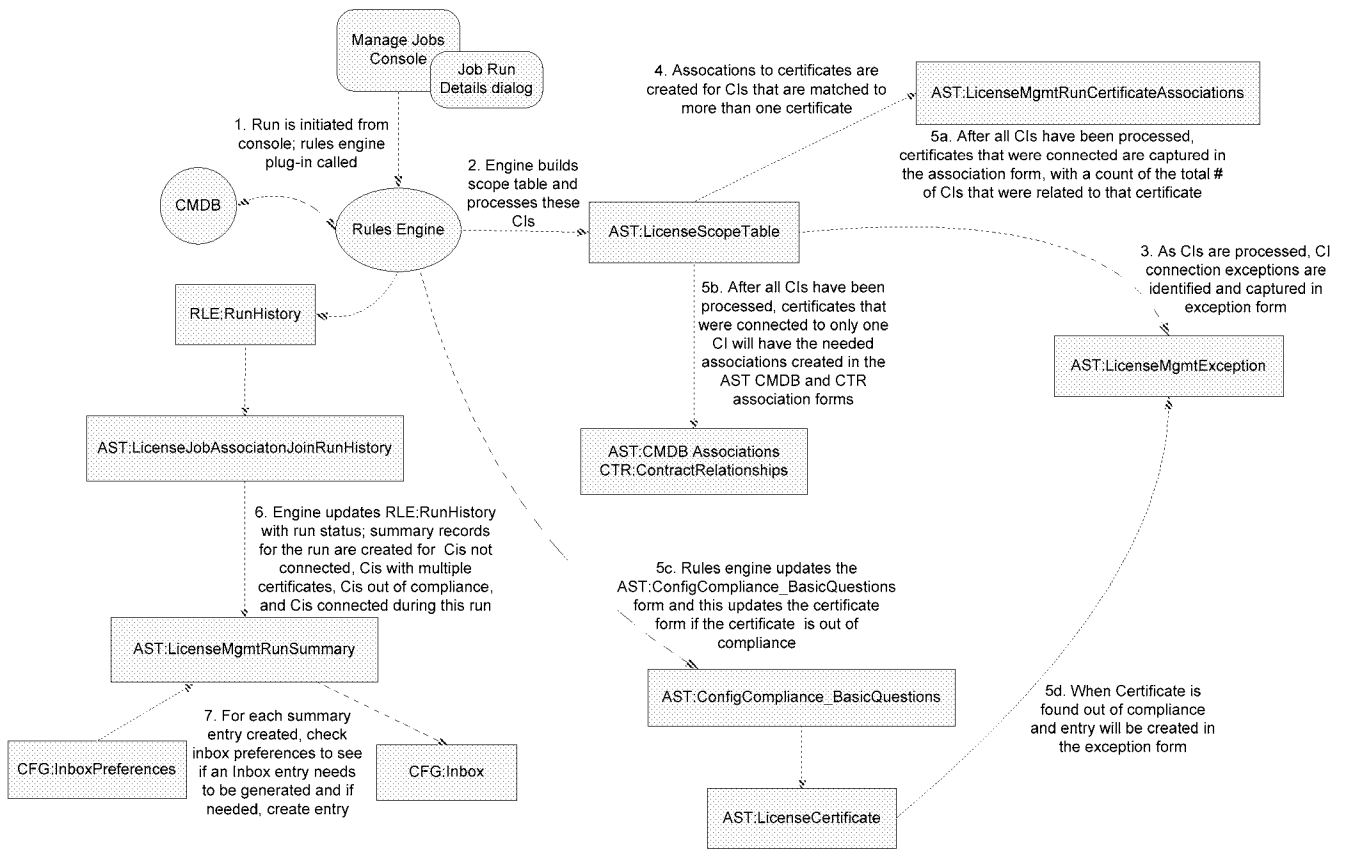
CIs that are successfully connected to certificates will also cause data to be captured in a certificate association form so that we know which certificates were affected, and the number of CIs that were connected to each certificate.

Once the rules engine has completed its processing, 4 summary records for the run will be produced for these cases...CIs not connected, CIs with multiple certificates, Certificates out of Compliance and Certificates connected to CIs. Based on the summary information, Inbox entries will be generated based on the Inbox Preferences.

In addition, complete details of the run will be captured which will include

- a list of all CIs that were not connected to a License Certificate
- a list of CIs that were identified to match to multiple License Certificates
- a list of License Certificates that went out of compliance due to the job run
- a list of License Certificates that were connected to CIs during the run, and the # of CIs connected to each certificate

2.2 Process Flow Diagram



2.3 User Interface

The results of a job run will be displayed on the “Manage License Jobs” console.

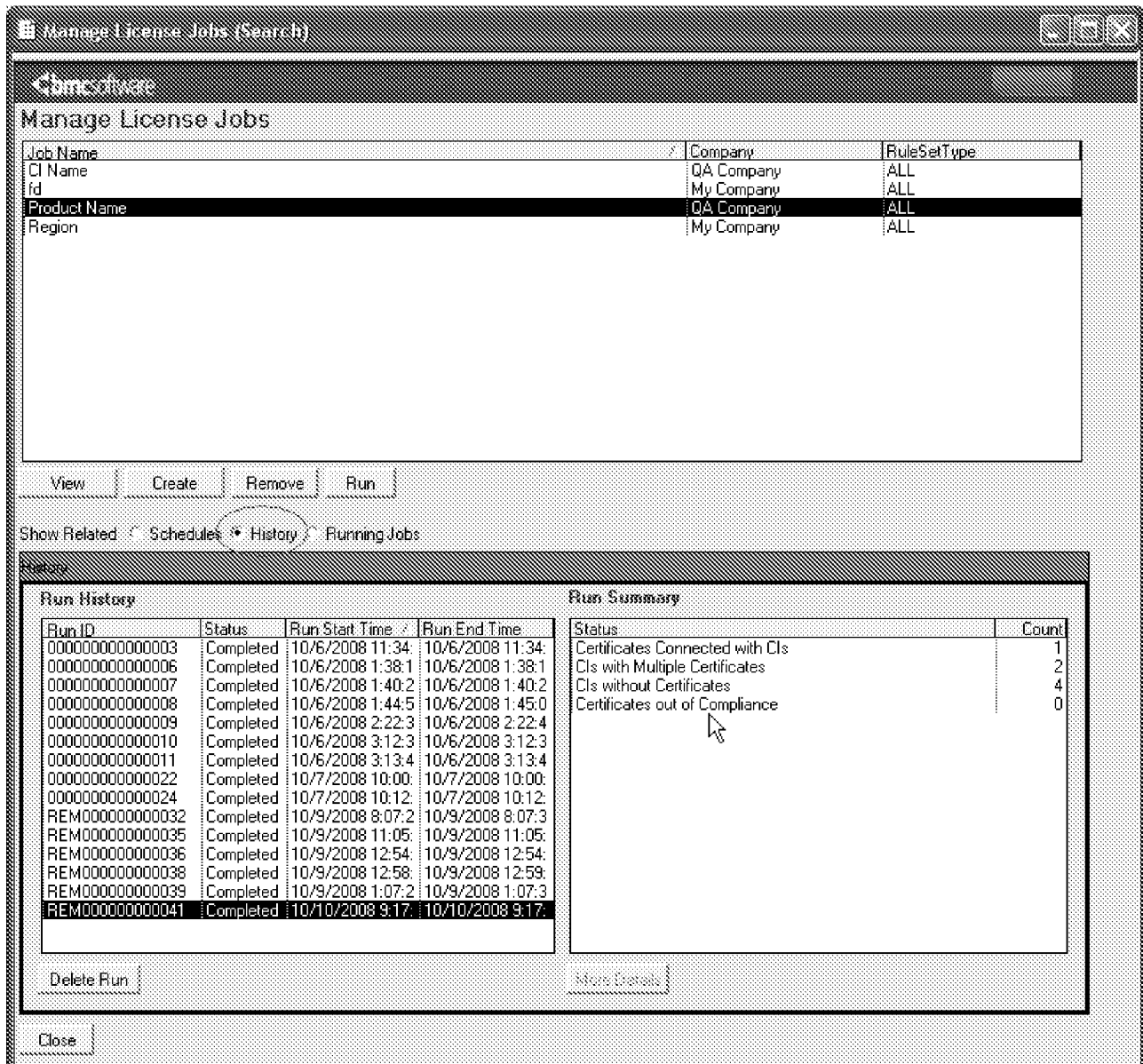


Figure 2.3.1 Manage License Jobs console with Run History panel displayed

The History panel of the Manage License Jobs console will contain 2 tables. The left table (Run History) will contain a list of all runs for the selected job. Run History records can be deleted from here by selecting the “Delete Run” button.

The right table (Run Summary) will summarize what occurred during the selected run. Each item in the Run Summary table can be selected, and if the count is greater than 0, you will be able to view additional details of that entry by selecting the “More Details” button. This will bring up the Job Run Details dialog (see figures below).

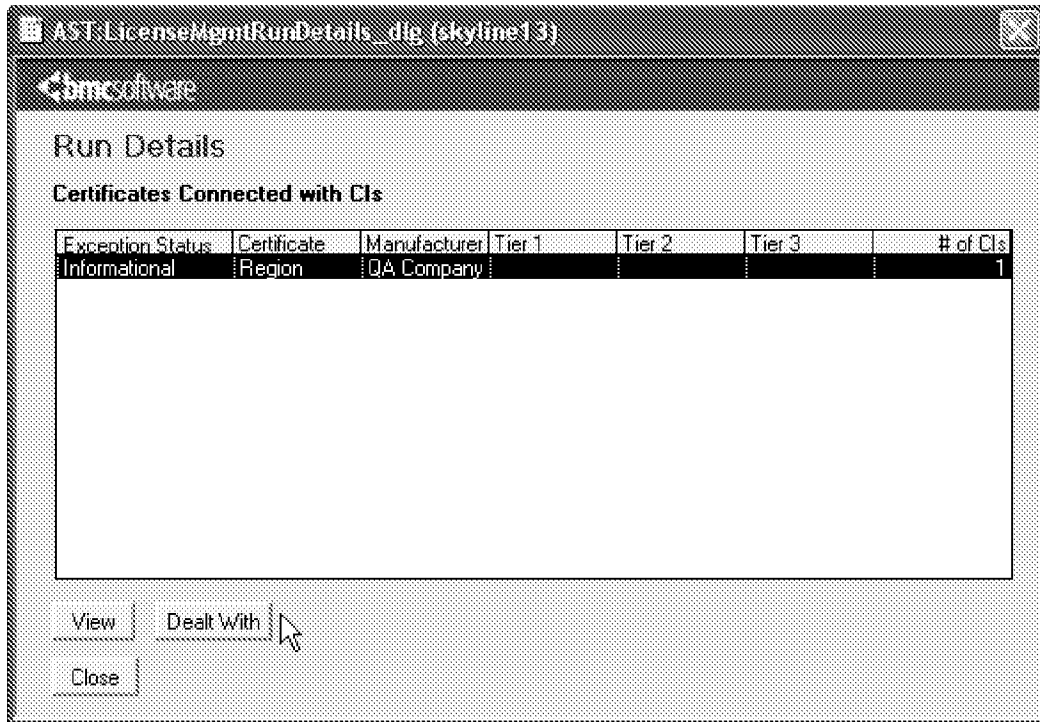


Figure 2.3.1a Job Run Details dialog – Certificates Connected with CIs view

The “Certificates Connected with CIs” view will show the certificates that were connected to CIs and the total number of CIs that were connected for this run. The “View” button will open up the details of the certificate so you can see which CIs are related to the certificate.

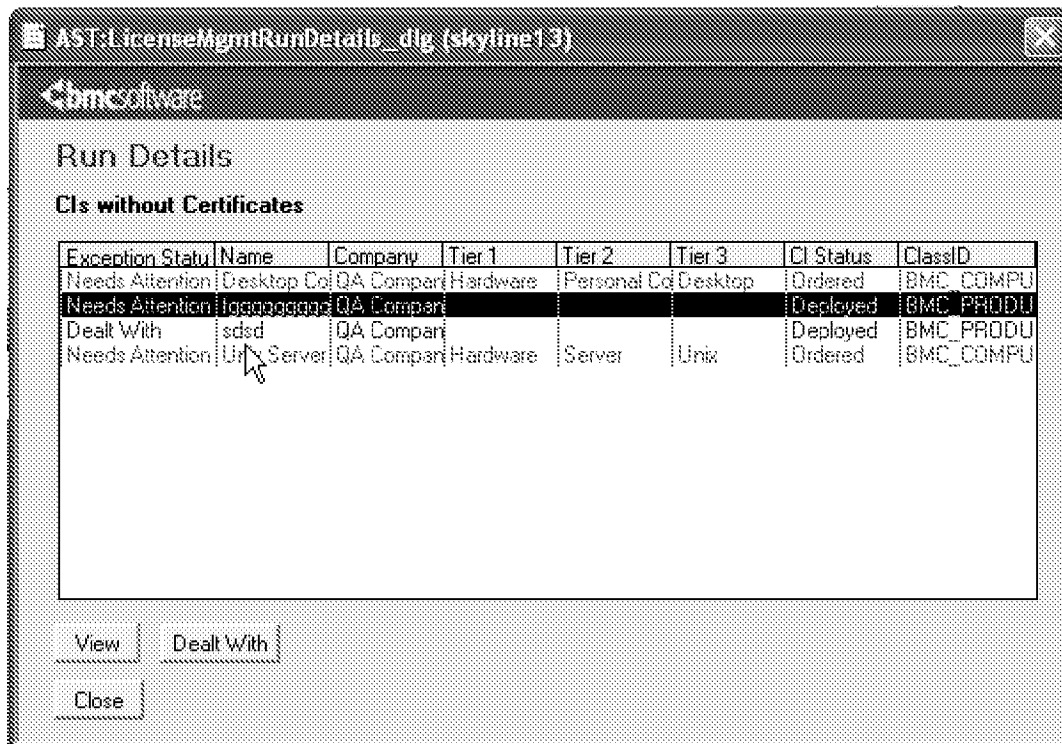


Figure 2.3.1b Job Run Details dialog – CIs without Certificates

The “CIs without Certificates” view (see figure 2.3.1b) will show you a list of CIs that did not get connected to any License Certificate. The “View” button will open up the details of the selected CI. The “Dealt With” button allows you to change the Exception status from “Needs Attention” to “Dealt With”.

The view for the “CIs with Multiple Certificates” details shows you a list of CIs that could be matched with multiple license certificates, as well as the certificates that matched. See figure 2.3.1c below.

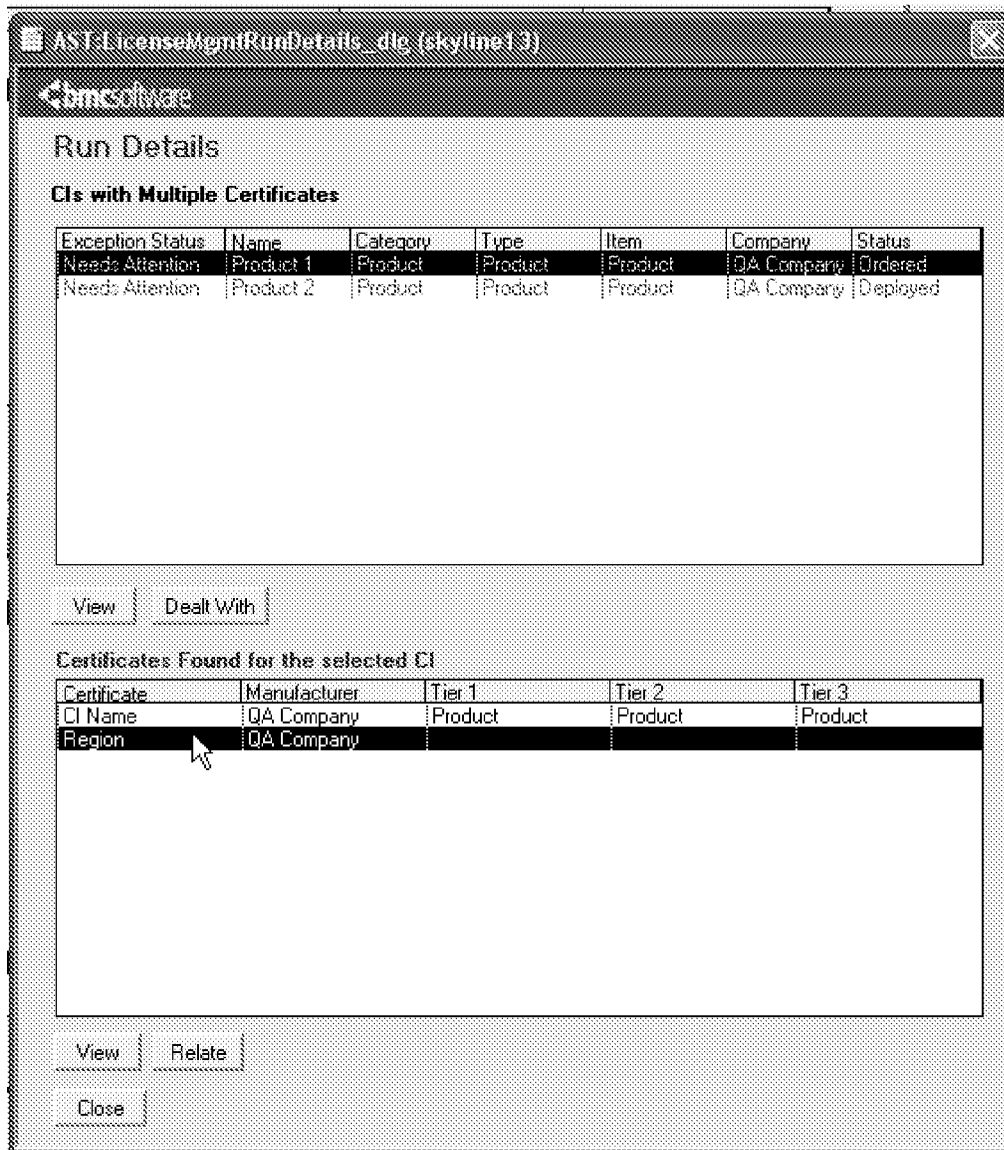


Figure 2.3.1c Job Run Details dialog – CIs with Multiple Certificates

Selecting the “Dealt With” button on this dialog will change the Exception Status from “Needs Attention” to “Dealt With”

Selecting “View” from the top table will open the details of the selected CI. Selecting “View” from the lower table will open the details of the certificate.

Selecting “Relate” will relate the selected CI from the top table to the selected certificate in the lower table, and will update the Exception Status for the CI to “Dealt With”.

You will also be able to view CIs that were not connected to a license certificate for the last run of a specified job via the SAM Console. From the console, you can select a job and see the CIs that were not related to any certificate for the last run for that job.

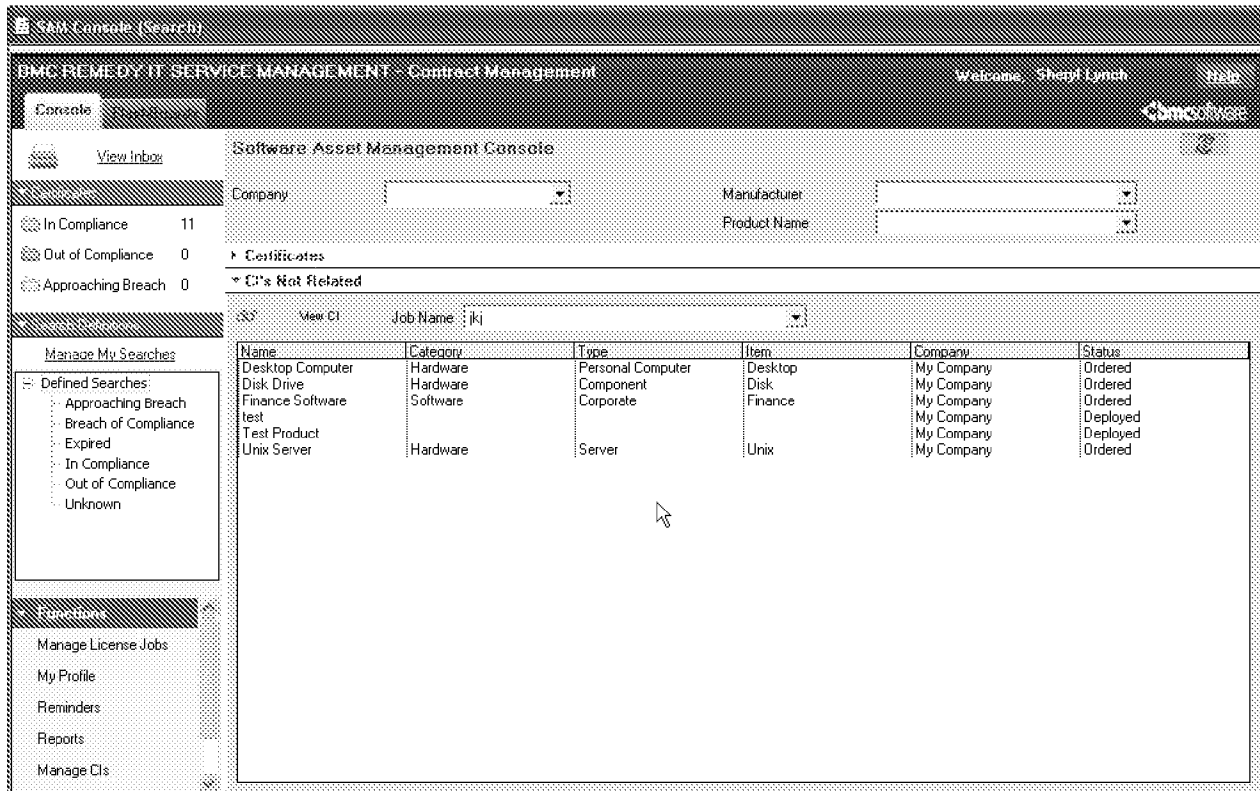


Figure 2.3.2 SAM Console – CIs Not Related

3.0 Solution, Product or Component Design

3.1 New/Modified Forms

AST:LicenseMgmtException form – existing form; fields added to support SWLM exceptions for 7.5 release;

AST:LicenseMgmtRunSummary form – holds summary data for each run

AST:LicenseMgmtRunCertificateAssociation form – holds association data for CIs with multiple certificates, and certificates that were connected for a particular run

AST:LicenseMgmtRunDetails_dlg – display only form to view run details

AST:LicenseMgmtExJoinJobAssocRunHistory – Join form; contains exception information for a specified job; used for other joins (see below)

AST:LicenseMgmtExJoinJobAssocRunHistoryJoinBaseElement – Join form; contains CI information for CIs with connection exceptions; used for Job Run Details dialog

AST:LicenseMgmtExJoinJobAssocRunHistoryJoinLicCertificates – Join form; contains certificate information for certificates that are out of compliance; used for Job Run Details dialog

AST:LicenseMgmtRunCertAssocJoinLicCertificates – Join form; contains certificate information for certificates that were connected during a run; used for Job Run Details dialog

3.2 Data

SYS:MenuItems – data used for Event Type and Request Type menus on Inbox Preferences form when configuring inbox preferences for license exceptions

Menu Type	Locale	Selection C.	Status	Menu Label 1	Menu Label 2	Menu Value 1
SWLMRunJobEventType		4000	Enabled	Certificates Connected with CIs	keyConnectedCerts	Certificates Connected with CIs
SWLMRunJobEventType		1000	Enabled	Certificates out of Compliance	keyCertOutOfCompliance	Certificates Out of Compliance
SWLMRunJobEventType		3000	Enabled	CIs with Multiple Certificates	keyCIsMultiCert	CIs with Multiple Certificates
SWLMRunJobEventType		2000	Enabled	CIs without Certificates	keyCIsNoCert	CIs without Certificates

3.3 Workflow Overview

The workflow listed below is a list of key workflow to support this feature.

Workflow to support deletions of run history data:

- Active Link : AST:LJC:RunHistory_101_DeleteRun; fires when “Delete Run” button is selected on Manage Jobs console; pushes z1D_Action = “DELETE” to RLE:RunHistory and AST:LicenseMgmtException forms; pushes ‘deleted’ = “true” to AST:LicenseMgmtRunSummary and AST:LicenseMgmtRunCertificateAssociaion forms
- Filter : RLE:REM>Delete_795_Delete Record; fires if ‘z1D_Action’ = “DELETE” on RLE:RunHistory form and deletes current record
- Filter : AST:SHR:LicenseMgmtRun_795_DeleteEntry – deletes record from AST:LicenseMgmtRunSummary and AST:LicenseMgmtRunCertificateAssociaion forms when ‘deleted’ = “true”
- AST:LEX>Delete_100_DeleteRecord – exsiting workflow that deletes record from AST:LicenseMgmtException form if ‘z1D_Action’ = “DELETE”

Workflow to trigger creation of summary records when Run has completed:

- Filter : AST:REM>CreateSummaryData; fires on modify of record in RLE:RunHistory form when Status > “Running”; pushes Short Description = “CREATESUMMARY” to AST:LicenseJobAssociationJoinRunHistory form

Workflow to create run summary information:

- 4 filters tied to form AST:LicenseJobAssociationJoinRunHistory
- Naming convention: AST:JRH:SetSummary_XXX
- These filters are triggered by Short Description = “CREATESUMMARY” and create entries in the AST:LicenseMgmtRunSummary form

Workflow to create exception from AST:LicenseCertificate form when certificate becomes out of compliance due to a job run :

- AST:ALC:ComplianceException_750_PLEX! – filter that creates an entry in the AST:LicenseMgmtException form if compliance status = “Out of Compliance” and Run id is specified.

Workflow to create exceptions from AST:LicenseScopeTable form

- AST:LST>CreateExceptions_700! – creates entry in AST:LicenseMgmtException form for CIs without

- certificates and CIs with multiple certificates (Count = 0 or Count > 1)
- Workflow to create certificate associations from AST:LicenseScopeTable form
- Filter AST:LST:SetCertificateID_555_CreateExceptionAssoc`! – creates certificate association for CIs with multiple certificates if Count > 1
- AST:LST:CreateExceptions_710_CreateAssoc`!—creates certificate associations for certificates that were connected during the run if Count = 1

Workflow to create Inbox entries

- 2 Filters on AST:LicenseMgmtRunSummary form that are triggered on Submit
- Naming convention: AST:JRS:Inbox_XXX
- Filters query CFG:Inbox Preferences to check for entries for that company (or company = “- Global –“ and Event Type, and if found, generates and entry in CFG:Inbox.

1.0 Solution, Product or Component Overview

The license type creation wizard allows for an administrator to create custom license types for license certificates. Once the question wizard is completed, a user creating certificates will be able to answer questions about the license type on the certificate form.

The purpose of the connection and compliance question building wizard is to:

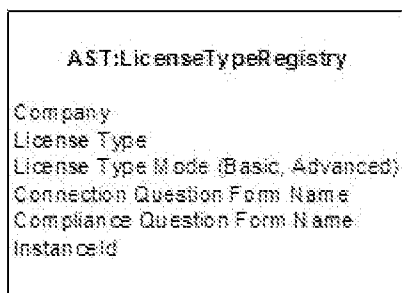
- 1) Give users a wizard like UI for creating custom license types,
- 2) Add questions for a particular type.
- 3) Define the mappings for what to query on, and what to do with the results.

2.0 Architecture

2.1 License Type Registry

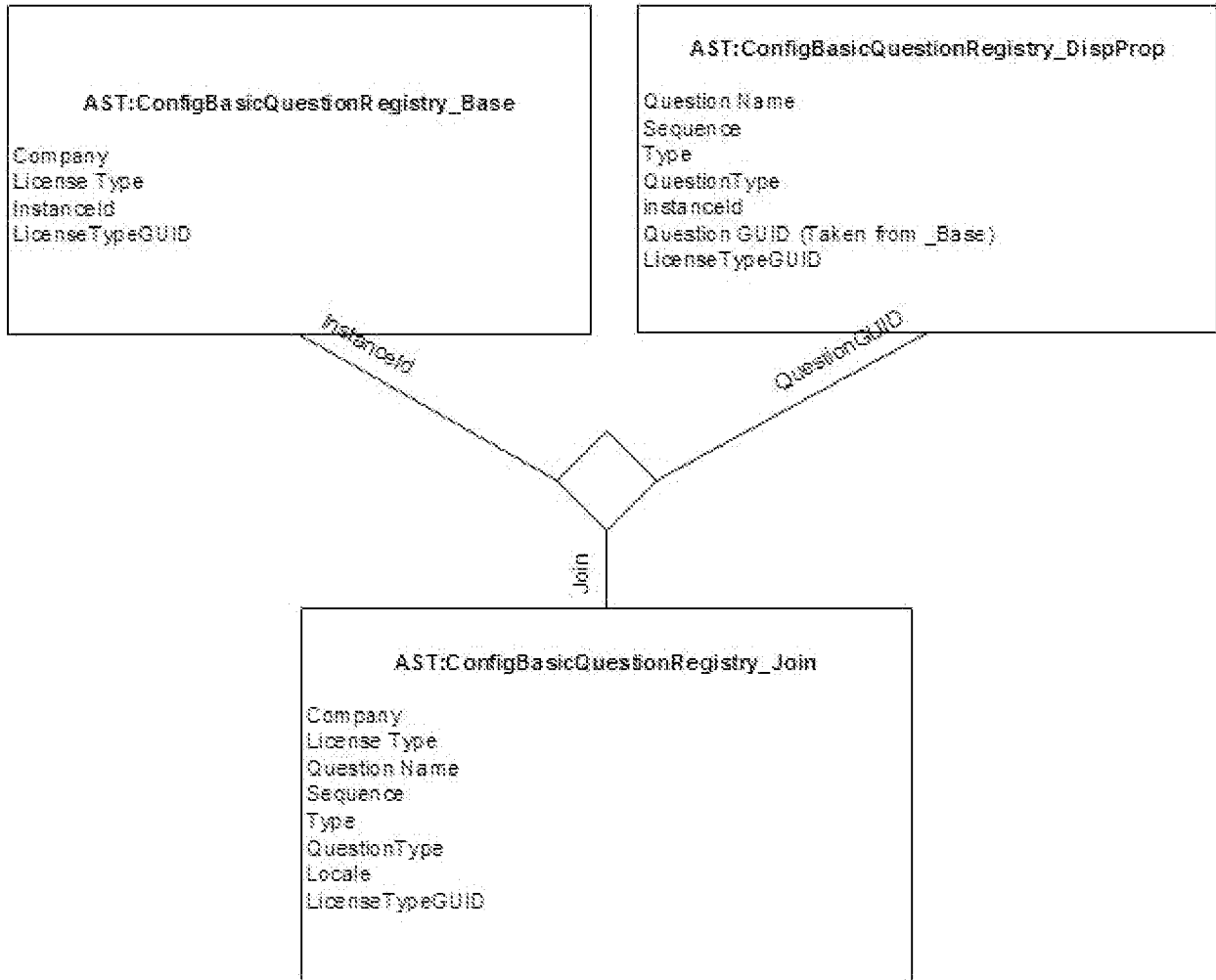
The registries store the metadata that will be used to interact with the engine. They will be populated when the User creates a license type. There will be 3 registry parts to hold the data:

- 1) AST:ConfigBasicQuestionRegistry_Base, AST:ConfigBasicQuestionRegistry_DisProp, AST:ConfigBasicQuestionRegistry_Join
 - This holds the created license types and their information. It is used as a menu on the AST:LicenseCertificates form.



2.2 Connection Questions and Mapping

- When a question is created, data is pushed to the <>_Base and <>_DispProps forms.



AST:ConfigConnection_RuleFieldMapping, AST:ConfigCompliance_RuleFieldMapping

- These forms are composed so that the user can enter questions for different locales. The Join form is where the complete dataset is stored.
- When a mapping is generated, the sequence of the question determines the value of the
- The form “AST:ConfigConnection_RuleFieldMapping” holds the source value (answer field, result of an operation) and the target value (CMDB attribute, AR field ID, ect) for each question. The value

```

AST:ConfigConnection_RulefieldMapping
LicenseTypeGUID
Connection Question Form Name
QuestionGUID
QuestionSource (Answer field)
QuestionTarget (AR field ID)
Source
Class Name
Attribute Name

```

```

AST:ConfigConnection_RulefieldMapping
LicenseTypeGUID
Compliance Question Form Name
QuestionGUID
QuestionSource
QuestionTarget
Source

```

2.3 Compliance Questions and Actions

- 1) The Questions will be stored in the Question Registry. They will be displayed on the License Certificate form.
 - When a question is created, a reference of that question will be stored in the AST:ConfigCompliance_RuleReference form. This will allow actions created to reference a compliance question
- 2) The questions will be used in the Compliance Actions forms.
 - There are 4 types of actions
 - **Get:** This will get a list of values or a count of the matching records on a form
 - **Compare/Calculate:** This will calculate or compare an expression
 - **Update:** This will update a list of values on a form
 - The action metadata will be stored in a form (AST:RuleDefinition). This form will have all of the required information to push to the RLE:<>RuleInterface forms.
 - When get/calculate/compare action is created, a reference for the result will be created in the AST:ConfigCompliance_RuleReference form. Actions will be able other action results that were created before the current action.
 - A table reference on the AST:ConfigRuleSet will reference the AST:RuleDefinition form.

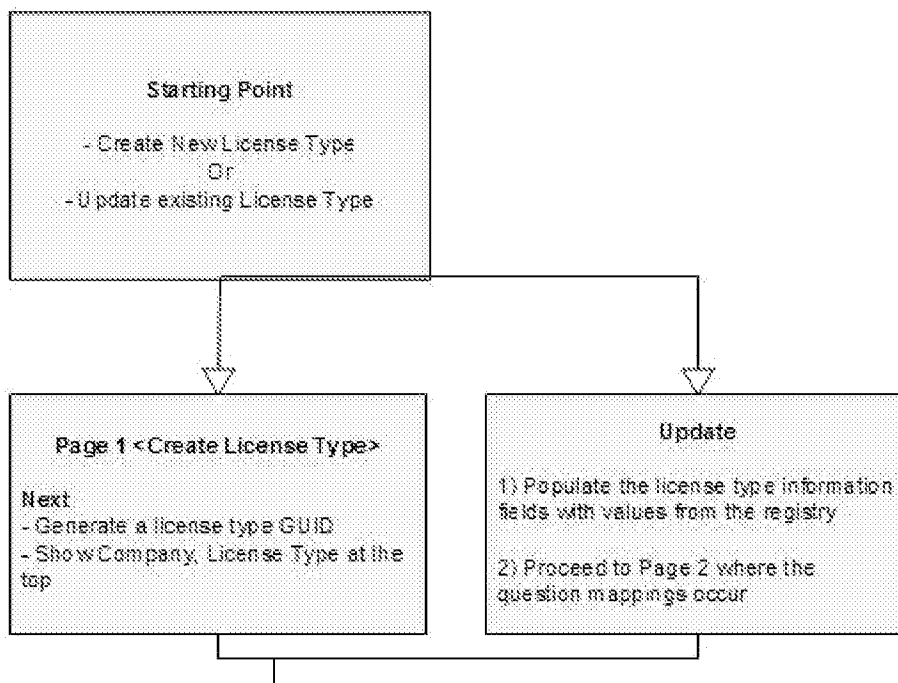
2.4 Wizard Flow

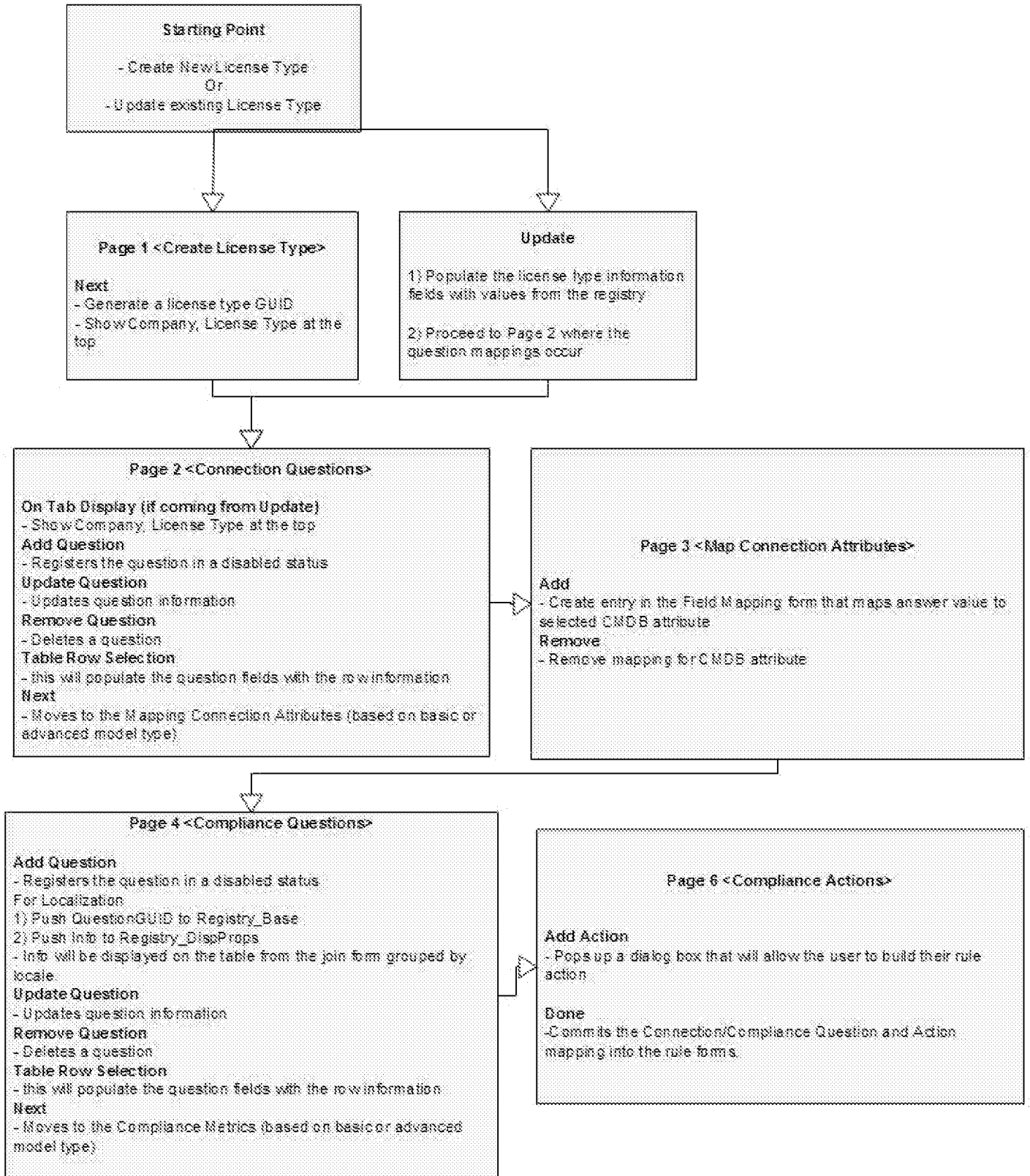
Below is a diagram of the process flow for a user to create a license type.

- 1) The User will be prompted to create or update a license type. If they choose to create a new license type, they will be brought to aS page that will allow them to enter in the appropriate information. If they choose to

update an existing license, they will be brought to the connection question page.

- 2) Once the user has entered the basic license type information, they are brought to a series of question and mapping forms.
 - a. **Connection Data:** This question set is responsible for returning a list of CIs that match certain criteria (based on the answers given to the questions). Field mappings between the answer fields and CMDB attributes to query are defined.
 - b. **Compliance Data:** Data here will perform compliance operations based on the actions defined and the returned list of CIs.
- 3) Once all of the information is completed, the user will be brought to a summary page to confirm their choices. They will then be given the option to generate the defined rules for the license type. Once the data is committed, workflow will engage the engine to build the appropriate rule set.





2.5 Interaction with the Engine Interface

When the user has completed out the questions and mapping and commits to build the rules, the rules get generated by the inputted data being pushed to the engine. (NOTE: View the Engine SDD on the SWLM sharepoint for engine interface inputs).

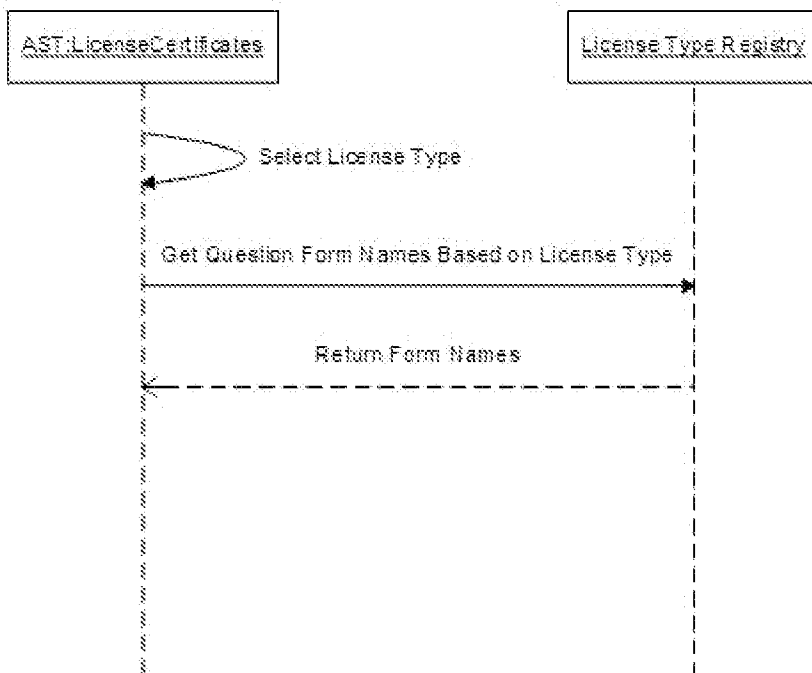
Once triggered on commit, there will be a set of filters (separated by guides) that will push out to the engine interface forms.

3.0 Solution, Product or Component Design

User Interaction from the Certificate form:

3.1 Selecting a License Type on the CTR:LicenseCertificates form

When a user selects a license type, a call is made to the CTR:LicenseTypeRegistry form to get the names of the question forms to open. The purpose of this is so that the user can create their own question forms based on the license type they select.



3.2 Answering the questions

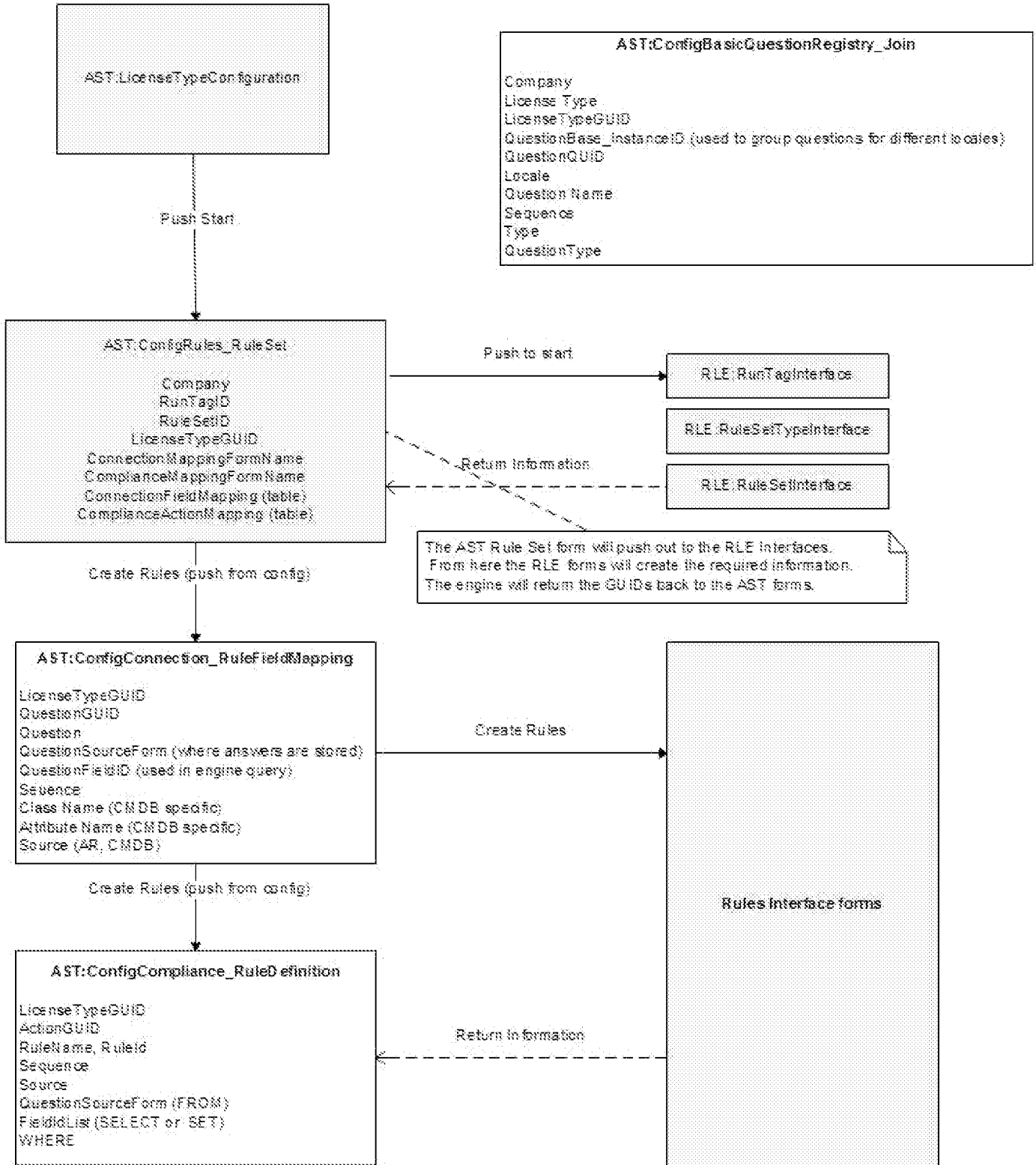
When the user answers the questions, the answers will be stored on a data form. When the engine is run, it will retrieve the answers defined.

4.0 Interaction with the Rules Engine

The contract administrator will create the rules mapping using a rule creation mechanism (a form dialog). The rule building dialog will consist of:

Field	Description
License Type	Rules will be defined for a particular license type.
Rule Name	This will be the name of the rule
Attribute	This will be a menu of CMDB attributes used to search for a CI.
Field	This will be a menu that has a list of fields from the question forms. The menu on this field will be driven by the license type that is selected. ***** There may be multiple attributes per rule. Should there be a table of some sort to store multiple attributes?
Form Name	This will be a hidden field that will be set when a license type is selected (it will be gotten from the license type registry). The engine will use this value to retrieve the field information for the rule qualification.

- Once the rule is saved, the information will be pushed to an interface form (CTR:ConnectionQuestions_<Lic Type>Interface/CTR:ComplianceQuestions_<LicType>Interface).
 - o ***** The answer mapping may be pushed to the rule interface forms.
- When the engine is run it will create the connection rules based on the answers and mappings.
- Once the rule is created, the engine will query the CMDB for a list of CIs that match the connection information.
- Once the CIs are gathered and the appropriate information is gathered from them (through connection rules). The gathered data will be stored on a run-time form.
- The engine will then run the compliance rules on the list of CIs using the rules defined for that license type.
- The engine will output the events (exceptions, messages) to the event module.



1.0 Solution, Product or Component Overview

1.1 Business Requirement

This document will define all existing business rules in ITSM version 7.0 for the Contract type Software License.

1.2 Functionality Description

1.2.1 Definition of attributes for AST:LicenseCertificates form

Attribute Name (Label)	DB Name	Comments
Status	Status	New/Current/Obsolete/To Be Deleted
ID	CertificateName	Identifier for Certificate
Summary	CertificateDescription	Summary of Certificate
Contract ID	Contract ID	Foreign Key from Contract.
License Category Type	LicenseCategoryType	Selection field containing the following values: Client/Server/Mainframe
License Type GUID	LicenseTypeGUID	Reference to License Type of the certificate
License Type	z1D_LicenseType	Localized value of License Type. Obtained using License Type GUID
Connection Form Name	z1D_ConnectionFormName	Connection Form Name based on the License Type. Obtained using License Type GUID
Compliance Form Name	z1D_ComplianceFormName	Compliance Form Name based on the License Type. Obtained using License Type GUID
License Restriction Level	RestrictionLevel	*READ ONLY*? Selection field containing the following values: Enterprise Level, Site Level, Fixed Number
License Effective Date	Effective Date	
License Expiry Date	Expiry Date	Entry will be blank if license does not expire

Attribute Name (Label)	DB Name	Comments
License Unlimited Flag	Unlimited Flag	Entry will be checked if the license does not expire.
License Notification Date	Notification Date	New
Master Certificate ID	MasterCertificateInstanceID	Field to determine what the Master Certificate's instance ID will be.
Certificate Grouping Flag	GroupCertificateFlag	Selection Field containing attributes: No Group, Attached to Group, Master Certificate. Display only?
Certificate Sequence	CertificateSequence	Number to determine the sequencing of certificates in a group. Only applies to certificates that are attached to a group.
Compliance Information	CertificateComplianceStatus	Selection field containing attributes: In Compliance, Out of Compliance, Unknown. Needs to be a Read only field, determines if certificate is in compliance or not.
Certificate Owner Company		Required field
Certificate Owner Support Organization		Required field
Certificate Owner Support Group Name		Required field
Certificate Owner		Not a required field
Product Categorization Tier 1		
Product Categorization Tier2		
Product Categorization Tier 3		
Product Name		
Manufacturer		Required Field
Product Model/Version		

Attribute Name (Label)	DB Name	Comments
Purchase Cost		Display only field linked to Costs table
Purchase Cost Instance ID	PurchaseCostInstanceID	Holds instance ID of cost entry.
Cost Center		Pulled from Software Contract, needed for costs.
Cost Per Asset		
Compliance Fields		Fields that will contain compliance questions and fields to answer these questions. These will be display-only and will be called from the Compliance Forms.
Connection Fields		Fields that will contain connection questions and fields to answer these questions. These will be display-only and will be called from the Connection forms.
Purchase Line Item Instance ID	PurchaseLineItemInstanceID	Holds Instance ID of Purchase Line Item.
Order ID	z1D_PurchaseOrderID	Temp field that identifies Order ID, derived from Line Item Instance ID
Line Item	z1D_PartNumber	Temp Field that identifies Part Number derived from Line Item Instance ID

1.2.2 Definition of workflow for AST:LicenseCertificates form

Attribute Name (Label)	Details	Comments
Save		Number of Licenses, Licenses in Use and Licenses available are updating the attributes on the Software Contract form
Search Product Category		Use AST:ProductDictionarySearch to query the PDL:Product Dictionary form

Attribute Name (Label)	Details	Comments
Group Certificate		After entering Product Categorization, check to see if an existing certificate exists for this certificate based on Product Categorization. If there is, need to ask user if we want to “group” the certificates together.
Upgrade/Renew		Upgrade/Renew License Certificates by creating a copy of the existing certificate and declaring the original as obsolete.
Wizard Next and Previous Buttons		Buttons will hide and unhide certain page fields while receiving information from the user. At Page 2 (Product Dictionary Entry), the next button will trigger workflow that checks for grouping.
Unlimited License		If the License has no expiry date, check “License Unlimited Flag” and grey out expiry date.
Show Terms and Conditions		Show Terms and Conditions of Certificate.
Show Rights Granted		Show Rights Granted of Certificate.
Obtain Compliance Questions		Obtain data pertaining to Compliance Questions from Compliance Form
Save Compliance Answers		Push data pertaining to Compliance answers to Compliance Form
Obtain Connection Questions		Obtain data pertaining to Connection Questions from Connections Form
Save Connection Questions		Push data pertaining to Connection answers to Connections Form.
Purchase Cost	Temp Field for Purchase Cost	If Purchase Cost field is filled in, create purchase cost entry with its instance ID stored on the certificate. Likewise, remove entry when Purchase Cost field is no longer filled in.

Attribute Name (Label)	Details	Comments
Purchase Line Item	Temp Fields for Purchase Order ID and Line Item	User can enter a purchase line item entry by entering the Purchase Order and Line Item, it will show an error if these do not match. Likewise, can search for a line item for this certificate, the related instance ID of the Line Item will be stored on the certificate as a reference. If Order ID and Line Item are blank, the reference is deleted.
Create Software Asset	Relationships with CIs	User can create a CI to identify a relationship with a certificate.
Relate Software Asset	Relationships with CIs	User can search for a CI to identify a relationship with a certificate.

1.2.3 Definition of attributes for AST:GroupCertificateDlg

Attribute Name (Label)	DB Name	Comments
Product Categorization Tier 1		Search Criteria for Master and Ungrouped Certificates
Product Categorization Tier 2		Search Criteria for Master and Ungrouped Certificates
Product Categorization Tier 3		Search Criteria for Master and Ungrouped Certificates
Product Name		Search Criteria for Master and Ungrouped Certificates
Model/Version		Search Criteria for Master and Ungrouped Certificates
Manufacturer		Search Criteria for Master and Ungrouped Certificates
Master Certificate ID		Holds the Master Certificate Instance ID which will be used to link ungrouped certificates to the master.

1.2.4 Definition of workflow for AST:GroupCertificateDlg

Attribute Name (Label)	Details	Comments
Available Groups and Ungrouped Certificates	Table	Uses Product Categorization to check for available Master Certificates or any ungrouped certificates. This table is only active when initiated from an ungrouped certificate
Child Certificate Table	Table	Depending on the selection of the Available Groups and Ungrouped certificates, shows table of its child certificates if a group is selected.
Ungrouped Certificates Table	Table	Shows all ungrouped certificates for the chosen Product Categorization. Only active when initiated from a Master Certificate
Select Certificate	Button	Selects chosen master to use to group selected ungrouped certificates.
Add to Group	Button	Add ungrouped certificates to a Master Certificate.

1.2.5 Definition of attributes for SHRAAS:SHR-SearchPanel form

Attribute Name (Label)	DB Name	Comments
Certificate ID	Certificate Name	Search by Certificate ID
Contract ID	Contract ID	Search by associated Contract ID
License Expiry Date Low Range	ExpiryDateLow	Search by License Expiry date
License Expiry Date High Range	ExpiryDateHigh	Search by License Expiry date
License Unlimited Flag	Unlimited Flag	Search for all licenses which do not expire.
Product Categorization Tier 1		
Product Categorization Tier 2		
Product Categorization Tier 3		
Product Name		
Manufacturer		
Product Model/Version		

1.2.6 Definition of workflow for SHRAAS:SHR-SearchPanel form

Attribute Name (Label)	Details	Comments
Certificates Table	Reference to AST:LicenseCertificates	Columns include Certificate ID, Software Module, Version Number, License Type, License Category, Effective Date, Expiry Date, Notification Date, and certificate owner info. Validation of table based on info entered into search.
Relate		Relate selected certificate to CI.
Relationship Type		For certificates, the relationship types will be Upgrade Of, Related To, and Renewal Of.
View		View selected certificate
Close		Exit Dialog without relating certificate.

1.2.7 Definition of workflow for Contracts Tab on Contract Forms

Attribute Name (Label)	Details	Comments
Contract Table		Table Property will point to CTR:Associations. Columns will include Contract ID, Contract Type, Association Type, Contract Description, Contract status, Notification Date, and Expiry date.
Contract Type		Possible Contract Types: Parent, Child, Master, Related, Certificate
Add		Change Functionality to allow user to choose a contract to relate.
Create		This will allow user to create a new contract that will be related

1.2.8 Definition of workflow for AST:ProductDictionarySearch form

Attribute Name (Label)	Details	Comments
Add		Use AST:ProductDictionarySearch to query the PDL:Product Dictionary form
Note	PDL:ProductDictionary	Join form PDL:ProductModelVersion PCT:ProductCompanyAssociation
Note	PDL:ProductModelVersion	Join form PCT:Product Catalog PCT:Product Model/Version
Note	PDL:ProductDictionary	Displays only Product where a Model/Version has been created
Remove Workflow	AST:PDR:RelatePD_008_Che ckDup	No longer needed
Remove Workflow	AST:PDR:RelatePD_015_Dup Found	No longer needed

1.2.9 Definition of attributes for FIN:Payments (formerly CTR:Contract Payments) form

Attribute Name (Label)	DB Name	Comments
Parent ID*	Parent ID	This field is formerly Contract ID, with this form moving to FIN subsystem, should be used as a parent ID for any payment entry.
Payment Code	Payment Code	
Payment Period	Payment Period	
Payment Check Number	Payment Check	
Date Payment Due	Date Payment Due	
Date Payment Sent	Date Payment Sent	
Payment Amount Due	Payment Due	

Attribute Name (Label)	DB Name	Comments
Payment Amount Sent	Payment Sent	
Payment Type		
Cost Reference		

1.2.10 Definition of workflow for CTR:Contract Payment form

Attribute Name (Label)	Details	Comments
Workflow		Any workflow present on this form should now be prefixed with FIN:ATX: as it is now part of FIN subsystem.

1.2.11 Definition of attributes for FIN:Costs

Attribute Name (Label)	Details	Comments
Unit Type	The current values of this field are Flat Rate, Minutes, and Hours, all other values are taken from menu of Payment Period field on CTR:ContractPayments	Selection field to include the following: Flat Rate, Minutes, Hours, Days, Weekly, Bi-Weekly, Monthly, Quarterly, Semi-Annual, Annual, Bi-Annual, Other
Cost Type		Menu Field. May need to swap with menu SYS:MNU:Contracts-PaymentCode when Costs form is opened from Contract or add configuration data to FIN:ConfigCostType
Cost Category		Menu Field. May need to add data to FIN:ConfigCostCategory to ensure that contract types are included.

1.2.12 Definition of Workflow of Financials Tab for Certificate

Attribute Name (Label)	Details	Comments
Add Purchase Cost		If Purchase cost has been entered, create a new FIN:Costs entry. If Purchase cost has modified with an existing FIN:Costs entry, push data into appropriate FIN:Costs Record.
Delete Purchase Cost		If Purchase cost has been removed, remove associated FIN:Costs entry.

1.2.13 Definition of Workflow of Financials Tab for Contract

Attribute Name (Label)	Details	Comments
Financials Tab		Renamed from Payments Tab.
Payments Table	I think we need to separate Costs from Payments and thus make Payments a separate form.	Table will read from FIN:Payments (formerly CTR:ContractPayments).
Costs Table		On Software Contract, will roll up costs related to purchase costs of certificates. On all other contracts, will roll up purchase costs related to the Contract.
Total Payments Made		Active Link Guide will roll up all Contract Payments
Total Purchase Cost		Active Link Guide will roll up all Purchase Costs
Add Payment		Open FIN:Payments form to create new payment entry
View Payment		Open FIN:Payments form to view Payment entry.
Delete Payment		Delete Payment Entry
View Cost		Open FIN:Costs form to view Purchase Cost Entry.

Attribute Name (Label)	Details	Comments
Add Cost		Add Cost entry. Can only add costs in all contracts except Software License.
Delete Cost		Delete Cost entry. Can only delete costs in all contracts except Software License.

1.2.14 Definition of Attributes of AST:PurchasingSearch_dlg

Attribute Name (Label)	Details	Comments
SearchRequisition	View	View designed to search for Purchase Requisitions
SearchOrder	View	View designed to search for Purchase Orders
SearchLineItem	View	New View designed to search for Purchase Line Items
Order ID		Search Criteria for Line Items
Requisition ID		Search Criteria for Line Items
Date Ordered		Search Criteria for Line Items
Status		Search Criteria for Line Items
Part Number		Search Criteria for Line Items
Company		Search Criteria for Line Items
Product Categorization Tier 1		Search Criteria for Line Items
Product Categorization Tier 2		Search Criteria for Line Items
Product Categorization Tier 3		Search Criteria for Line Items
Product Name		Search Criteria for Line Items
Model/Version		Search Criteria for Line Items

Attribute Name (Label)	Details	Comments
Manufacturer		Search Criteria for Line Items
zTmpSearchCriteria		Field for External qualifier for POLineItems_tbl table.

1.2.15 Definition of Workflow of AST:PurchasingSearch_dlg

Attribute Name (Label)	Details	Comments
POLineItems_tbl	Table	Table to show search results based on external qualifier
Search Button		Triggers table refresh based on external qualifier
Clear Button		Clear all search criteria
Relate		Relate selected line item to parent.

1.2.16 Definition of Attributes of AST:ConfigCompliance_BasicQuestions

Attribute Name (Label)	DB Name	Comments
Question Label	QuestionLabel[1-10]	10 fields, one for each question, Shows the question to be asked.
Question Type	QuestionType[1-10]	10 fields, one for each question, determines if question requires an integer or character answer.
Character Answer	Ans_Character[1-10]	10 fields, one for each question, holds character answers (if applicable)
Integer Answer	Ans_Integer[1-10]	10 fields, one for each question, holds integer answers (if applicable)
Integer Answer Type	IntQuestionType[1-10]	10 fields, one for each question, for each integer question, checks to see if it is summable, distributable, or neither.

Attribute Name (Label)	DB Name	Comments
Summable Question Identifier	SummableQuestionGUID[1-10]	10 fields, one for each question, for each distributable integer question, holds a reference to the summable question.
Compliance Alarm	ComplianceAlarm[1-10]	10 fields, one for each question, for each distributable integer question, check to see if compliance will be breached if a distributable question violates a summable question
Status	Status	Only Status values Draft and Executed will be used in rollup calculations.
Master Certificate ID	MasterCertificateInstanceID	Similar to AST:LicenseCertificates, allows child runtime entries to determine which is the master runtime entry.
Certificate Compliance	CertificateComplianceStatus	Selection field containing attributes: In Compliance, Out of Compliance, Unknown. Needs to be a Read only field, determines if certificate is in compliance or not.

1.2.17 Definition of Workflow of AST:ConfigCompliance_BasicQuestions

Attribute Name (Label)	Details	Comments
Group Compliance	Filter action 'zID Action' = "GROUP COMPLIANCE"	Sends data to Master Runtime to add the child runtime to group.
Ungroup Compliance	Filter action 'zID Action' = "UNGROUP COMPLIANCE"	Removes runtime data from group, reinitializes rollup to check if still in compliance.
Update Compliance	Master runtime filter action 'zID Action' = "UPDATE COMPLIANCE"	Updates master runtime with questions from child runtime. This is not the same as a rollup.

Attribute Name (Label)	Details	Comments
Rollup Compliance	Filter action 'z1D Action' = "ROLLUP COMPLIANCE"	When distributable data is entered in the runtime, this condition needs to be triggered in order to properly distribute the licenses amongst its child runtime entries. If the distributable value is greater than the summable value, the certificate is out of compliance.
Compliance Runtime Table	Table z2TH_ComplianceRuntime	This table is accessed by master runtime entries and is used by filter workflow to sum up summable data. It is also used in filter guides to walk table values to distribute distributable values.
AST:RollupCompliance_DistribInt	Filter Guide	Filter Guide used to walk entries on z2TH_ComplianceRuntime to determine distribution of distributable values.

1.3 Functionality Use Case

N/A

1.3.1 Use Case Diagram

N/A

2.0 Architecture

2.1.1 AST:LicenseCertificates

Use Case Scenario:

- Open Contract Console
- Create SWLM Contract and enter details
- Create a certificate as a child from the contract
- Certificate will flow user through answering questions based on type
- Kick off engine to do initial run for the certificate
- Engine processes connections and compliance and generates events
- Events are evaluated in the Contract/SWLM console

To relate a PO to a certificate or to relate a PDE to certificate, we require a form to allow us to store certificates. The AST:Keys and Versions form is considered to be antiquated and will be replaced by a new form called AST:License Certificates. We will require the following functionality to go with this form:

- Use of Product Categorization: We require the use of Product Categorization fields to allow us to relate a certificate to a Product Dictionary Entry.
- License Category Type: the out of box values will be Client, Server, and Mainframe
- License Certificate Type: the out of box value will be "Per Instance".
- License Restriction Level: this determines whether a License is constrained by location or number. Out of box values include Enterprise Level, Site Level, and Fixed Number.
- Product Dictionary Grouping: If similar entries are entered into Product Categorization fields, allow the user to "group" these entries so that they fall under the same certificate.
- Search Certificate: In order to relate a certificate to a Line Item, Contract, or PDE, we require a new Certificate Search form.
- Foreign key relationship with Contracts. Contracts will have a one-to-many relationship with License Certificates.
- Dates: Effective Date, Notification Date, and Expiry date will be needed. The notification Date will be used to notify the certificate owner, whether it be a group or individual. Also, we need to take into account "non-expiring certificates", that is, certificates without an expiry date. A flag will be used to identify these certificates, Notification Date and Expiry Date will be blank.
- Renewal or Upgrade: when the certificate is renewed or upgraded, a copy of the new certificate is made. The original certificate is declared obsolete.
- Terms and Conditions
- Rights Granted.

Integration with the Software License Engine:

- View Field: This view field would be required as a "window" to view certain forms. In this case, it would be used to view compliance information as well as answer questions related to the certificate so that the information can be used in the Software License Engine. (NOTE: If functionality of View field not compatible with our goals, we will revert to using Tabs instead).
- Compliance and Connection Questions: The wizard will use the license type defined to determine the questions that will be asked to determine Compliance and Connection. Also, the user has the option to re-answer the connection and compliance questions again when the Compliance Details and Connection details buttons are pressed.

License Certificate Wizard

Use case scenario

- We want to use a "wizard" to expedite the process of creating a certificate.

Flow

- From Contract, the workflow should open License Certificate in Wizard Dialog View.
- The wizard needs to display four separate "sections":
 - o 1) General Information: this includes information such as License Type, Compliance Info, Vendor and Supplier Companies, Dates, Certificate owners, costs, and Purchase Line Item information.
 - o 2) Product Dictionary: this allows user to enter product categorization info. It is here where the grouping of certificates is initiated.
 - o 3) Connection Details: the details for connection are entered here
 - o 4) Compliance Details: the details for compliance are entered here.
 - o 5) Summary: details summarizing the creation of a certificate are displayed here before the certificate is saved.
- After all information is entered, the user should be able to save the certificate. If the user has selected to group certificates, then the certificate should be attached to the group.
- The certificate should show up in the Contracts table.

Grouping of Certificates

Use case scenario

- A license Certificate with a certain product categorization is entered.
- A second license certificate with the same product categorization as the first is entered.
- When the second certificate is saved, user should have the option to group the two certificates.

Concepts:

- Any Product Dictionary Entry can have more than one group
- A master certificate is created to “represent” the group.
- Two dialogs are required, one to display available master certificates (with table containing child certificates of master), and another to display any available ungrouped certificates.

Flow

- Foreign key relationship is created to Master certificate (i.e. a field will be on the certificate form to identify a certificate’s master.
- Field GroupCertificateFlag will be used to identify if a certificate is part of a group, not part of a group, or a master certificate.
- Workflow to group certificates should come during screen two of the wizard.
 - o Check to see if there are certificates that match the PDE of the certificate being entered.
 - o If certificates with similar PDE exist, prompt a message to the user stating so. User has the option to group these certificates.
 - Check to see if a master certificate exists, if it does, show the Master Certificates dialog and prompt the user if he wants to:
 - Add the certificate to a selected group represented by a Master Certificate.
 - Create a new Master certificate.
 - If a master certificate is selected, take the Master Certificate ID for reference
 - Check to see if there are any ungrouped certificates. Show a dialog if these ungrouped certificates exist.
 - If ungrouped certificates are selected along with a master certificate, add this certificate along with others to the selected group represented by the master certificate.
 - If ungrouped certificates are selected with the user previously opting to create a new master, create a new group and a new master certificate.
 - If no ungrouped certificates are detected or selected and master certificate was previously selected, add this certificate to the selected group represented by the master certificate.
 - If no ungrouped certificates are detected or selected and the user previously opted to create a new certificate, take no action.

Master Certificate

Use Case Scenario

- See Grouping of Certificates. When certificates with similar Product Dictionary Entries are created, a master certificate is created. This certificate will represent all certificates under its group.
- There can be more than one Master Certificate per product dictionary entry.
- The Master Certificate will serve as a “rollup of licenses”, that is, connection and compliance information will act as a summary to all of its child certificates, it would have the sum of all license information held by its certificates.

Flow

- Master Certificate is created by creating a group of certificates.
- Master certificates can be identified by Software Contracts, opening them will allow the user to view all child certificates.
- Master Certificates should not be subject to renewal. This should allow child certificates to be renewed without affecting the master.

Deletion of Certificates, Ungrouping

Use Case Scenario

- Certificates can be unattached to Cis as either the Certificate or CI age.
- May be possible to “ungroup” certificates.

Upgrade Considerations:

- The Form AST:Keys and Versions should be used as template for upgrade to 7.5. The form will not be superseded but rather data inside it should be used to create certificates.

2.1.2 License Type

Use Case: In certificates, we have a field called License Type which we use to ask certain “questions” and obtain certain “rules” about how to process the certificates into the Software License Engine. The information is pulled from the questions and rules and “pushed” into the Engine.

Requirements

- Templates for generating the appropriate question and runtime forms
- Administer console interface for generating the rules
- SWLM Friendly interface for generating the rules
- SWLM engine interface for storing the rules

Flow:

- Open Administrator Tool (AR)
- Copy Template Question Form for new type (AR)
- Copy Template Business form for new type (AR)
- Open Application Administrator Console (AM Console)
- Register New License Type (AM Console)
- Register forms for New License Type (AM Console)
- Open Rules UI form to build rule set (Contract Specific UI to Engine Rules)
- Interface with Rules Interface Form or License Type (SWLM Engine Functionality)

Description of the flow

As License Type will be a selection-field data-driven menu, this means that the data for License Type field will be stored in a configuration form (Propose a separate form to do this, we need License Types to be configurable depending on the customer, we also need to “match” questions and rules with the appropriate License Type). Each License type will be configured with the appropriate Connection Questions and Compliance Questions forms. When the information in the menu is retrieved, information regarding these forms will also be retrieved.

At present, only one License Type is created out-of-box: Per Instance. The user can create his own license types to be used in the application. While creating the license type, they can create compliance and connection questions to be used. Both compliance and connection questions can be answered as integers or characters. For compliance type, integer questions can be either be Summable, Distributable, or neither. The use of summable and distributable questions can determine if a certificate is in compliance. (see 2.1.3 for more details)

It should be noted that certificate grouping will be dependant not only on the product categorization entered, but also the license type as well. Different license types can be used with the same product categorizations in different certificates, these certificates cannot be grouped.

While the information of the certificate is completed, the user will be asked questions regarding the certificate depending on what License Type is asked. This may come in the form of a “generic” questions form which outputs certain questions for the user to “answer” in predefined fields. Once these questions are answered, rules are pulled from the database (depending on the answers). The Certificate and these rules are then pushed to the Software License Engine to be processed.

Results of the Engine processing are likely to be included in an information table located on the SWLM console.

Dependencies:

2.1.3 Certificate Runtime

Use Case Scenario:

- The SWLM Engine can determine how many CIs are connected to a particular license certificate or group. We need to determine if this number violates data determining how many CIs can be allocated to a license and therefore if so, is this certificate out of compliance?
- If the license certificate is represented by a group, how do we “distribute” the number of CIs connected to the master certificate?
- What happens should a certificate is added to or removed from a group?

Concepts:

- Compliance Questions can be answered in character or integer format.
- There are three types of integer questions, Summable, Distributable, or None.
 - Summable questions involve a cap of the maximum allowable amount for a distributable property. These are answered by the user.
 - Distributable questions involve a value that is matched up against a summable question. If the value of the distributable question is higher than the summable question, the certificate may be out of compliance. This is further determined by whether this violation will trigger a compliance alarm.
 - Questions that are not summable or distributable have no weight in runtime calculations.
- On master certificates and integer questions:
 - The values given to summable questions on its child certificates are summed up at the master level.
 - The values given to distributable questions are collected at the master level and “distributed” amongst its child certificates. The amount given to a child certificate is determined by the cap given in the summable value.

Flow:

- Rollup of Runtime for each distributable question:
 - Determine its summable question.
 - Check to see if the distributable value is greater than the summable question.
 - If distributable value is greater than the summable value, check to see if a compliance alarm is to be triggered.
 - If the question requires triggering a compliance alarm, the certificate is out of compliance.
- Rollup of Runtime for each distributable question on a master level.
 - Allocate in order of certificate sequence as defined at certificate level.
 - Determine its summable question.
 - For each child certificate, determine what the summable value is.
 - For each child certificate, allocate the value equivalent or less than the summable value for that child certificate. Subtract this from the total distributable.
 - If all distributable values for its child certificates have been allocated and there is a remaining distributable value, it is likely the last certificate in the sequence that is out of compliance.

2.1.4 Ability to create a PO line item for a Certificate (1 cert to 1 Line item)

Examples:

- ability to tie a License Certificate to purchased software

Ability to relate a PO line item for a Certificate can be done by anyone with Purchasing user permissions, with any Contract permission except Viewer. Users with Contract User access can only access the certificates that belong to his group.

This is a simple reference relationship in which a Certificate will receive a foreign key relationship with a Line Item. Given that this relationship will allow for 1 certificate to one line item, no relationship form would be required.

We require a new view for AST:PurchasingSearch_dlg form which would be handled to search for Line Items only. Existing workflow that already searches for Purchase Requisitions and Purchase Orders will be leveraged to run the view to search for Purchase Line Items.

From Certificate side, the user will be able to see the associated Order ID as well as the Part Number of the Line Item. The user can either search for a new line item to relate to, view the current line item, or remove the relationship between the line item and Certificate.

Dependencies

Functionality affected (other)

Advantages

Disadvantages

Implications

Upgrades

Risks

2.1.5 Relationships with Contracts and Licenses

Use Cases:

- Enable different contract types to be related to each other.
- Enable a relationship to a contract that is not parent-child.
- Enable a relationship between a CI and a Certificate.

Ability to relate different contracts or contracts that are not of parent-child relationships can be done by anyone with a Contract Permission that is not viewer. Users with Contract User permission can only relate contracts that belong to their group.

Due to the complexity of the proposed SWLM and contracts model, we require the use of a new associations form for the use of CTR. This form will be referenced by the following forms via a Table:

AST:AssetSoftware
AST:AssetLease
AST:AssetMaintenance
AST:AssetWarranty
AST:AssetSupport
AST:LicenseCertificates
Master Contracts

The proposed use of this form is for the following:

- to relate contracts to each other (i.e. a support contract to a warranty contract, a support contract to another support contract)
- to relate contracts to a master contract
- to relate child contracts to a parent contract and vice versa. Note: Parent Contract ID may no longer be needed.
- To relate certificates to each other

- FUTURE: Need to separate Contract Relationships with CI and place them into this table as well. This will require substantial modifications to existing workflow to allow this to happen.

The CTR:Contract_Relationships form will be a “clone” of AST:CMDB Associations, and will have field attributes similar to the AST form. See table 1.2.11 for more details.

Currently, existing AST forms have a “Contracts” tab that contains a table that includes all child contracts attached to the parent. We want to replace that table with a table that points to the new CTR:Contract_Relationships form. On the certificate side, a new tab containing a table and appropriate buttons will be required.

This table will display all the contracts that are related to the current contract, regardless if the connection is through a straight relationship, parent-child relationship, or master contract. A selection field may be necessary to filter the types of contracts available, and to create or relate contracts based on the value of the selection field.

The table should show the following columns:

- ID of contract
- Description of Contract
- Type of Contract (Warranty, Maintenance, Software, Support, or Lease)
- Relationship of contract (parent-child, master, certificate, or straight relationship)
- Status of Contract
- Expiration Date
- Notification Date

Current functionality has the “Add” Button open the contract form in “New” mode with some of the fields on the contract copied from the original. The new contract is thereafter saved as the Child Contract of the original. While we want to maintain that functionality, we want to expand this functionality to include creating related contracts as well. Also, we want to include functionality that will allow us to create a parent-child relationship between two existing contracts.

The contract forms will use the existing form SHRAAS:SHR-SearchPanel which can search for contracts. However, we need to add functionality to identify what kind of relationship the two contracts will have. This will likely take place in the form of a selection field with the following values:

- Master
- Parent
- Child
- Regular Relationship
- Certificate

When a relationship is created, the data will be pushed into the CTR:Contract_Relationships form so that it will appear on the table in the Contracts tab as a Related Contract.

Functionality affected (other)

Advantages

Disadvantages

Implications

Upgrades

Upgrade workflow will be required to identify all parent-child relationships and create new association entries for each.

Driverscripts and views will be required to insert the new tables on each of the contract forms. A new view for form SHRAAS:SHR-SearchPanel will be required to insert the Contract Relationship Type field on the view.

Risks

2.1.6 Ability to create different (many) certificates for a PDE

Use case question

Flexible PDE attributes entry, certificates could be created with only categorization/product info.

We want to be able to identify any certificates by Manufacturer, name of software, and Version Number. Product categorization is optional.

Implementation question

A new field 'License category' is proposed based on Tony's email (scenarios).

The questions forms will capture if the license is for:Client (desktop), Server, Subscription, Mainframe...e.t.c. We will store the below 'License Types at the certificate level

- Desktop/client based licenses: (S5344)
 - o **per copy per device** (only one copy allowed on that device) (S5347)
 - example Office Professional 2003, Vista Operating System, Adobe Acrobat
 - capture # of licenses
 - details – found 1 license on computer – requires 1 license
 - details – found 2 licenses on computer – violation of license agreement (notification)
 - o **per device** (More than one copy can be installed on the licensed device) (S5348)
 - example Microsoft Office Enterprise 2007
 - capture # of licenses
 - details – found Office running twice on one computer – 1 licenses required
 - o **per copy** (one copy can be installed on multiple devices for a user) (S5349)
 - example is Adobe
 - capture # of copies
 - details (found 2 copies on separate machines owned by the same user) – 1 license is consumed
 - o **per instance** (any instance found requires a license)
 - example AR Server, BMC Remedy Asset Management Application
 - capture # of licenses
 - details – found 2 instances on computer – requires 2 licenses
 - Example to consider:
 - BMC Atrium CMDB Enterprise Manager is a per instance license.
 - However, the BMC Atrium CMDB Enterprise Manager – for enterprise is a site license
 - I would assume this would be a per-instance license type, with unlimited # of licenses restricted to a site.
 - o **per User** (requires a license for every user using the product – fixed license sometimes called named user)
 - example Microsoft Studio Development Addition 2008
 - capture # of users licensed
 - Could also capture tiers for costing
 - o **Per Concurrent user** (max number of users that can access the app at one time)
 - Example BMC Remedy Knowledge Management
 - Capture # of concurrent users
 - Could be unlimited

Description of the Flow

As the relationship between Product Dictionary entries to Certificates will be one to many, the Certificate form will have the Product Categorization entries on the AST:LicenseCertificates Form. The fields will be set up to take entries based on the Certificate Owner Company. That is, if the value of Certificate Owner Company is “Company A”, that means that there must exist a Product Categorization entry which has a company relationship with “Company A” as well as a Product Model Version. Such an entry is visible in the PDL:ProductDictionary join form. If there is no such entry for the Certificate Owner Company in this join form, then such a Product Dictionary entry does not exist, and as a result, this certificate cannot be created.

The order of the PDE fields should be as follows:

- Manufacturer (required)
- Product Name
- Model/Version
- Product Categorization Tiers 1-3.

Menus need to be revised so that all fields will be filled in in the order mentioned above. Upon Submit, workflow will check to see if there are similar certificates with similar Product Dictionary entries. If so, the workflow will ask if the certificates are to be grouped.

2.1.7 Contracts and Costs

Use Case: Implement costing module to contracts to capture costs

Please note that it is of the belief that Contract Payments are NOT the same as costs, as they identify payments towards a contract and not the actual costs of the contract. Any user with Contract permissions except viewer can access the Costing module.

At this time, certificates and costs will be integrated through the use of the Purchase Cost field. At present, the only planned link between Certificates and Costs will be made when the Purchase cost field is filled as a Cost entry will be created with the Certificate Purchase Price. On the Contract Level, a table will be used to roll up the costs of the certificates.

It is proposed that the relationship of Certificate and Cost can be identified in the following forms:

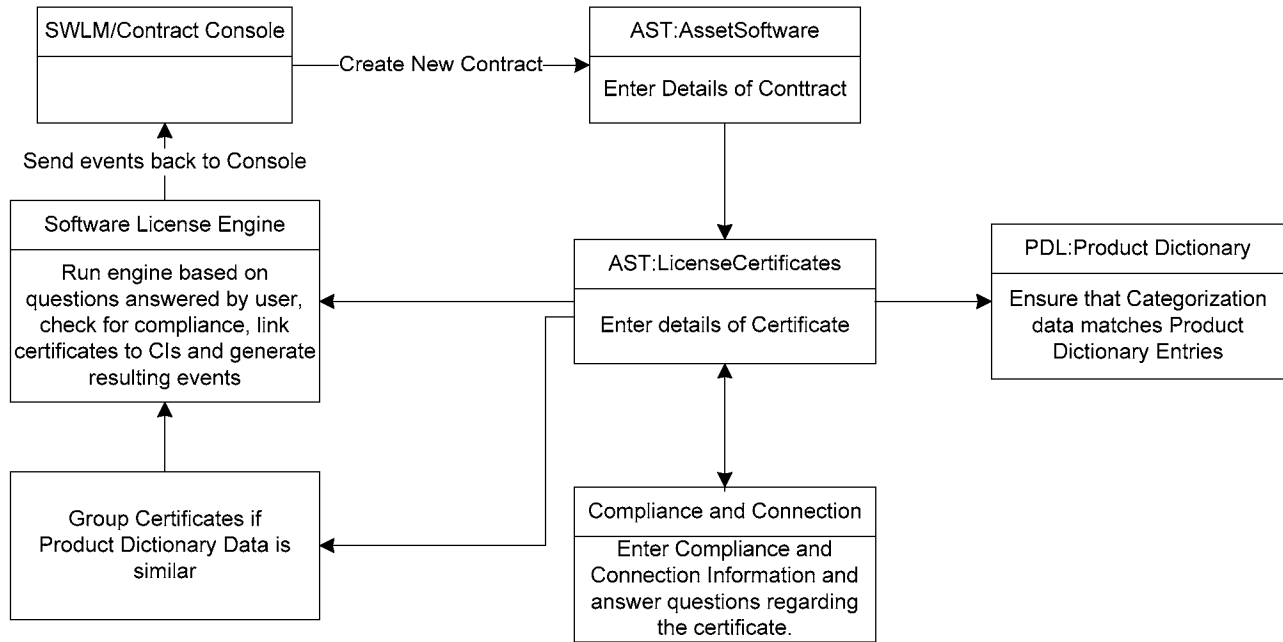
- **FIN:Costs**
 - Addition of data to Cost Category and Cost Type menus to accommodate Certificate Purchase price. Data will have to be added to FIN:ConfigCostCategory and FIN:ConfigCostType to support this.
- **AST:LicenseCertificates**
 - Purchase Cost field. When this field is entered, workflow should be allowed to capture the data and use it to create an entry in FIN:Costs. If the data in this field is modified, then the appropriate entry in FIN:Costs should be updated as well. If purchase cost is removed from certificate, the appropriate cost entry should be removed as well. The Purchase cost field should be display only, workflow will populate the field from the Cost entry.
- **Contract Forms**
 - CTR:Contract Payments should be shifted to FIN module
 - Payments Tab should be Financials Tab
 - A new table should be created which contains all the cost entries tied with the contract.
 - Software License Contracts will have a rollup of purchase cost entries. Users will not be allowed to add or remove cost entries from this form.
 - All other contracts will also have these cost entries, however they can be added or deleted from the Contract side.

This assumes that there are no other costs in place for Certificates, or contracts, and that Purchase Costs and Certificates enjoy a 1-1 relationship.

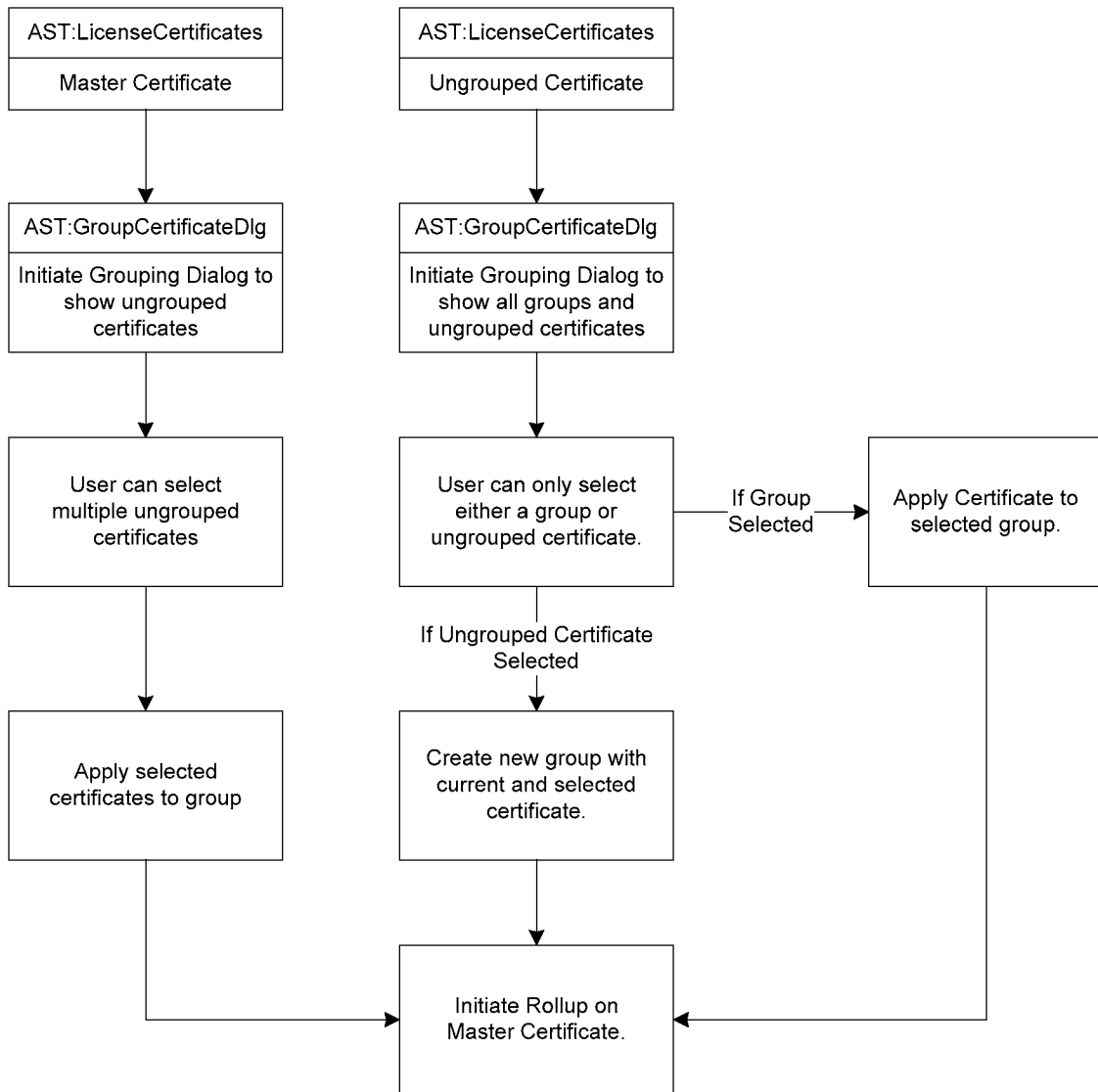
3.0 Solution, Product or Component Design

3.1 Diagrams

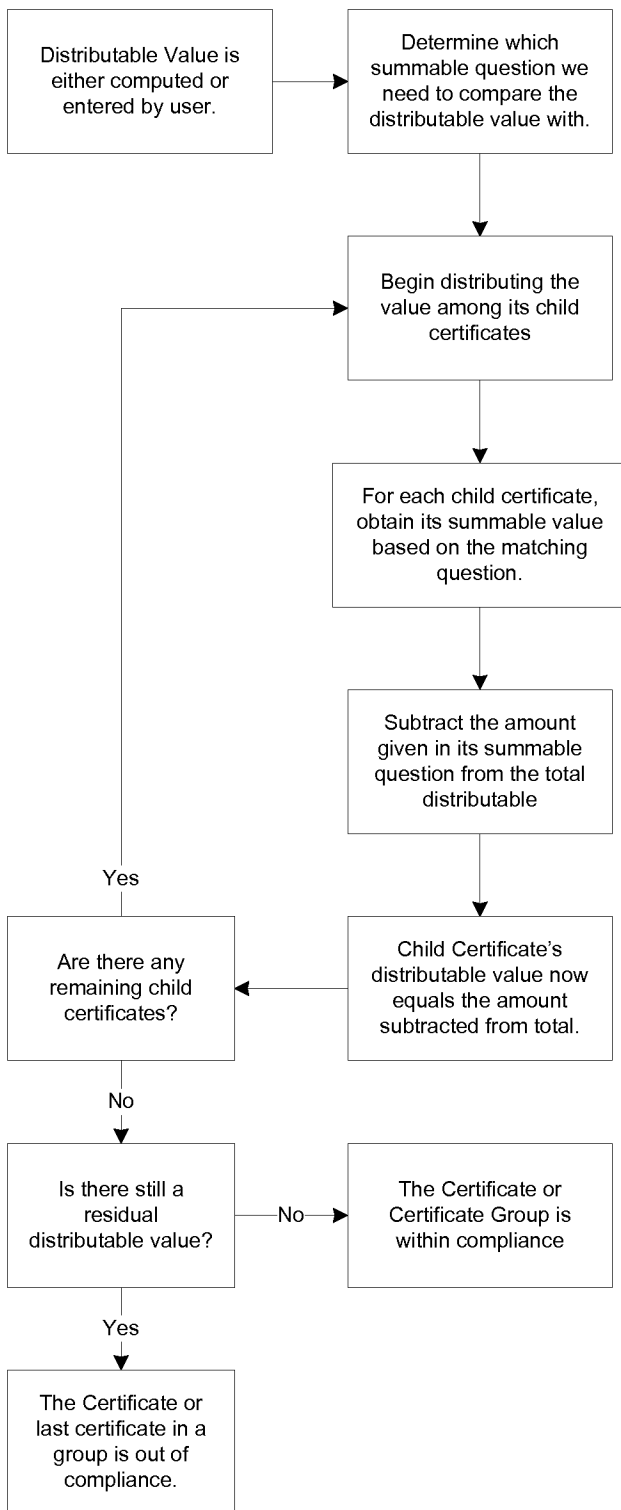
3.1.1 License Certificate, License Type,



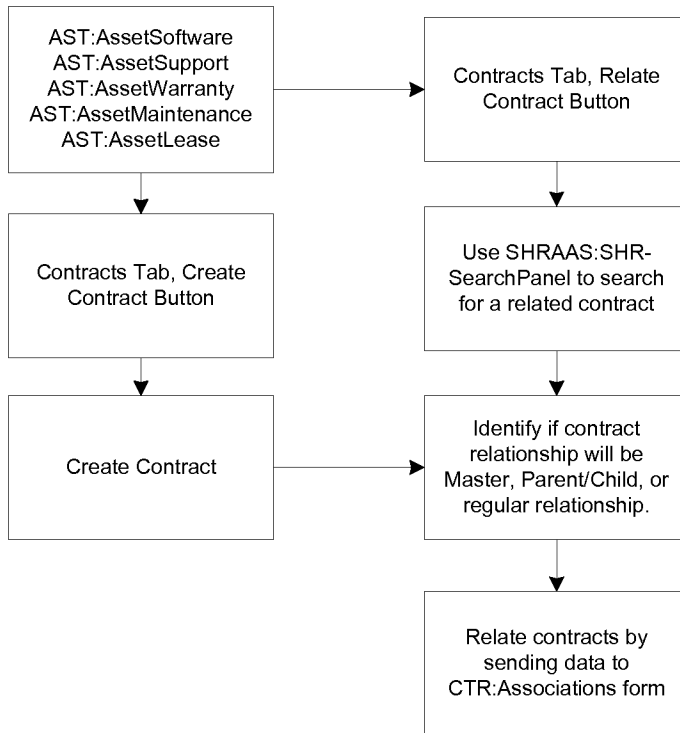
Grouping Flow



Runtime Flow:

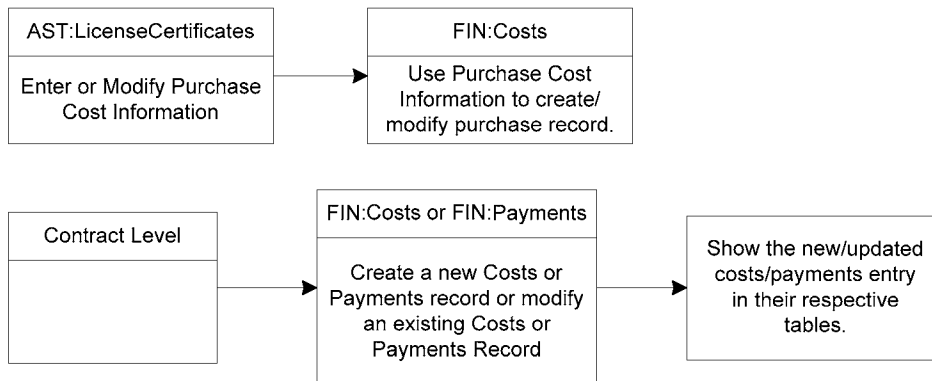


3.1.2 Contract to Contract Relationships



3.1.3 Contracts and Purchase Line Items

3.1.4 Certificates and Costs



Proposed Financials Mock-Up on Contract:

Cost Entries

Show



Date	Cost	Acquisition Type	Description
<i>Click to Refresh</i>			

Total Cost



Payments

Date Used	Date Sent	Payment Sent	Payment Date	Payment Period	Payment Code	Payment Description

1.0 Solution, Product or Component Overview

As part of the Software Asset Management Solution, Asset Management will have a wizard like UI that will support different types of Licenses.

We will have two models:

- 1) Basic
- 2) Advanced

The basic/Advanced models give users generic UI interface that can satisfy various licensing models. Asset Management will not only provide a model for creating your own license type, but will also provide you with some out of the box license types, pre-configured that can be used when defining Certificates:

These license types are as follows:

- 1) Per Instance
- 2) Enterprise License
- 3) Site License
- 4) Per Copy Per Device
- 5) Per Copy

2.0 Architecture

Out of the box we will ship the rules and data for the following License Types and here are the connection and compliance rules for each of these.

1. Per Company Or Enterprise

We will use the basic model to generate this:

Connection:

No questions for connection will be setup

Query BaseElement where Prod Cat Struct = Certificate.Prod Cat Struct and CI.Company = Certificate.Company AND @CIScope@

This list of CI's that will find a matching Certificate will be connected to the certificate

Compliance:

(No Compliance questions for this type)

There will be no Number of Licenses captured for this certificate,

The compliance rule would query Certificate form WHERE LicenseType = "Per Company" AND Company = Certificate.Company and Expiration Date > Current Date

If true Compliant else Not Compliant

2. Per Site

We will use the basic model to generate this, however need complex cmdb query support for this.

Connection:

1. Site

This will use the Complex query implementation in the engine.

The use will setup one question "Enter Site?" and this question will be tagged to Computer Systems as the Site that is used is the Site of the Computer System.

When the user clicks next, you go to the mapping screen that allows you to map the question to any Computer System Class.

There will be a complex query created for this as follows.. This is a combination of two queries.

1. Query Computer Systems(any of the class that is classified as computer system) where CS.Site = Certificate.Site AND Product.Prod Cat Struct = Certificate.Prod Cat Struct AND @CIScope@
2. Sub query within the first one would be .. Query Products, since there will be no direct questions for Product, so there is no mapping, we will query BaseElement WHERE Product.Prod Cat Struct = Certificate.Prod Cat Struct AND @CIScope@

This set of Products/CI's will be related to the Certificate

Compliance:

(No Compliance questions for this type)

This would be similar to Per Company

There will be no Number of Licenses captured for this certificate,

The compliance rule would include

1 Get Rule – To query Certificates WHERE LicenseType = "Per Site" AND @CertificateScope@ AND Expiration Date > Current Date AND Site = Certificate.Site

The CertificateScope parameter will include Certificate.Company = Company AND

Certificate.LicenseType=LicenseType AND Product Cat Struct = Certificate Product Cat Struct (any of those if defined for the run will also be used for qualifying)

3. Compare Rule to see if the number returned above is > 0

If true Compliant else Not Compliant

3. Per Copy Per Device

We will use the basic model to generate this:

Connection

No questions for connection

Query BaseElement where Prod Cat Struct = Certificate.Prod Cat Struct and CI.Company = Certificate.Company AND @CIScope@

This list of CI's that will find a matching Certificate will be connected to the certificate

Compliance

Following will be the compliance questions:

1. Enter Licenses purchased
2. Number of copies allowed per device

As part of computing compliance will have rules for computing the following:

1. Calculate Rule - Number Deployed (count of all the Software instances related to a Certificate)
2. Calculate Rule - Count of Computer Systems that have a product related more times than the <copies allowed per device>
3. Compare Rule - If #2 is true set the Certificate "Not Compliant"
4. Compare Rule - If Deployed > Purchased, set the Certificate "Not Compliant"

4. Per Instance

This is what we have currently in 7.0.x. We will use the basic model to generate this:

Connection:

No questions for connection

Query BaseElement where Prod Cat Struct = Certificate.Prod Cat Struct and CI.Company = Certificate.Company AND @CIScope@

This list of CI's that will find a matching Certificate will be connected to the certificate

Compliance

Following will be the compliance question:

1. Enter Licenses purchased .. user will enter from certificate
2. Number of Licenses Deployed .. this will be of type computed

As part of computing compliance will have rules for computing the following:

1. Calculate Rule - Number Deployed (count of all the Software instances related to the Certificate)
2. If Deployed > Purchased, Set "Not Compliant"

5. Per Copy

We will use the basic model to generate this:

Connection

No questions for connection

Query BaseElement WHERE Prod Cat = Certificate.Prod Cat AND Company = Certificate.Company AND @CIScope@

This set of Products/CI's will be related to the Certificate

Compliance

Following will be the compliance question:

1. Enter Licenses purchased

As part of computing compliance will have rules for computing the following:

1. Number of unique set of users connected to a product(Query API)
(Get List of Products for unique users)
Computers are related to People
Computers are related to Product
2. Count of Unique set of Users connected to Product > Purchased Then “ Not Compliant”

1.0 Solution, Product or Component Overview

As part of the Software Asset Management Solution, Asset Management will have a wizard like UI that will support different types of Licenses.

We will have two models:

- 1) Basic
- 2) Advanced

The basic/Advanced models give users generic UI interface that can satisfy various licensing models. Asset Management will not only provide a model for creating your own license type, but will also provide you with some out of the box license types, pre-configured that can be used when defining Certificates:

These license types are as follows:

- 1) Per Instance
- 2) Enterprise License
- 3) Site License
- 4) Per Copy Per Device
- 5) Per Copy

2.0 Architecture

Out of the box we will ship the rules and data for the following License Types and here are the connection and compliance rules for each of these.

1. Per Company Or Enterprise

We will use the basic model to generate this:

Connection:

No questions for connection will be setup

Query BaseElement where Prod Cat Struct = Certificate.Prod Cat Struct and CI.Company = Certificate.Company AND @CIScope@

This list of CI's that will find a matching Certificate will be connected to the certificate

Compliance:

(No Compliance questions for this type)

There will be no Number of Licenses captured for this certificate,

The compliance rule would query Certificate form WHERE LicenseType = "Per Company" AND Company = Certificate.Company and Expiration Date > Current Date

If true Compliant else Not Compliant

2. Per Site

We will use the basic model to generate this, however need complex cmdb query support for this.

Connection:

1. Site

This will use the Complex query implementation in the engine.

The use will setup one question "Enter Site?" and this question will be tagged to Computer Systems as the Site that is used is the Site of the Computer System.

When the user clicks next, you go to the mapping screen that allows you to map the question to any Computer System Class.

There will be a complex query created for this as follows.. This is a combination of two queries.

1. Query Computer Systems(any of the class that is classified as computer system) where CS.Site = Certificate.Site AND Product.Prod Cat Struct = Certificate.Prod Cat Struct AND @CIScope@
2. Sub query within the first one would be .. Query Products, since there will be no direct questions for Product, so there is no mapping, we will query BaseElement WHERE Product.Prod Cat Struct = Certificate.Prod Cat Struct AND @CIScope@

This set of Products/CI's will be related to the Certificate

Compliance:

(No Compliance questions for this type)

This would be similar to Per Company

There will be no Number of Licenses captured for this certificate,

The compliance rule would include

1 Get Rule – To query Certificates WHERE LicenseType = "Per Site" AND @CertificateScope@ AND Expiration Date > Current Date AND Site = Certificate.Site

The CertificateScope parameter will include Certificate.Company = Company AND

Certificate.LicenseType=LicenseType AND Product Cat Struct = Certificate Product Cat Struct (any of those if defined for the run will also be used for qualifying)

3. Compare Rule to see if the number returned above is > 0

If true Compliant else Not Compliant

3. Per Copy Per Device

We will use the basic model to generate this:

Connection

No questions for connection

Query BaseElement where Prod Cat Struct = Certificate.Prod Cat Struct and CI.Company = Certificate.Company AND @CIScope@

This list of CI's that will find a matching Certificate will be connected to the certificate

Compliance

Following will be the compliance questions:

1. Enter Licenses purchased
2. Number of copies allowed per device

As part of computing compliance will have rules for computing the following:

1. Calculate Rule - Number Deployed (count of all the Software instances related to a Certificate)
2. Calculate Rule - Count of Computer Systems that have a product related more times than the <copies allowed per device>
3. Compare Rule - If #2 is true set the Certificate "Not Compliant"
4. Compare Rule - If Deployed > Purchased, set the Certificate "Not Compliant"

4. Per Instance

This is what we have currently in 7.0.x. We will use the basic model to generate this:

Connection:

No questions for connection

Query BaseElement where Prod Cat Struct = Certificate.Prod Cat Struct and CI.Company = Certificate.Company AND @CIScope@

This list of CI's that will find a matching Certificate will be connected to the certificate

Compliance

Following will be the compliance question:

1. Enter Licenses purchased .. user will enter from certificate
2. Number of Licenses Deployed .. this will be of type computed

As part of computing compliance will have rules for computing the following:

1. Calculate Rule - Number Deployed (count of all the Software instances related to the Certificate)
2. If Deployed > Purchased, Set "Not Compliant"

5. Per Copy

We will use the basic model to generate this:

Connection

No questions for connection

Query BaseElement WHERE Prod Cat = Certificate.Prod Cat AND Company = Certificate.Company AND @CIScope@

This set of Products/CI's will be related to the Certificate

Compliance

Following will be the compliance question:

1. Enter Licenses purchased

As part of computing compliance will have rules for computing the following:

1. Number of unique set of users connected to a product(Query API)
(Get List of Products for unique users)
Computers are related to People
Computers are related to Product
2. Count of Unique set of Users connected to Product > Purchased Then “ Not Compliant”

1.0 Solution, Product or Component Overview

This functionality will provide a user interface for running license jobs.

There can be two types of runs that happen when engine is run, which are Connection and/or Compliance.

You can run the license jobs for either types:

- 1) ALL (Compliance and Connection)
- 2) COMPLIANCE

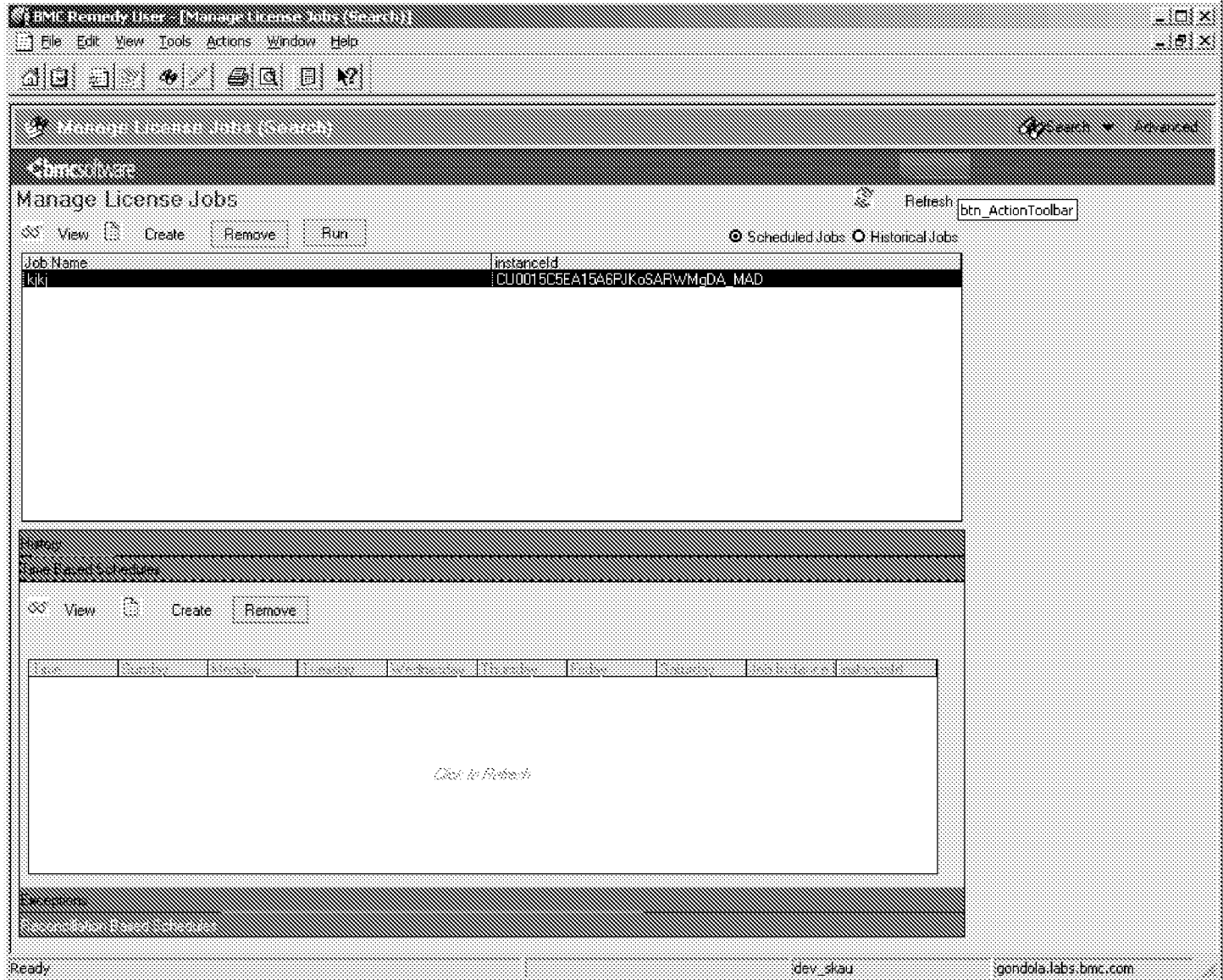
Connection will not be run separately, because whenever we create connections, we will have to follow it up with compliance type of runs as otherwise compliance would be outdated.

The AST:LicenseJob screen can be launched from the following places:

- 1) From the SAM Console, by clicking the Manage License job Link:
 - a. You can run existing scheduled jobs
 - b. You can schedule a job to be run at a specific time
 - c. You can select a job and run it
 - d. You can create a job and run it
- 2) From Certificates, whenever a certificate is modified, on save, a dialog pops open to say, do you want to run the license job at this time or schedule it for later, if yes, the create license job screen will open. Following changes will trigger this functionality:
 - a. Modification to questions and key fields (this should call both connection and compliance.)
 - b. If CI's are manually added or removed (this will call the compliance UI)

When the create license dialog opens from the Certificate, it will be populated with License Type, Product Cat, Company

Following is the screen shot of the Manage License Job dialog:



The show Jobs will have three choices:

- 1) Scheduled Jobs, it will be the default one
 - a. The bottom tabs will show you Time Based Schedules and Reconciliation Based Schedules, to see various times this jobs is scheduled for
- 2) Historical jobs
 - a. The bottom part will show two tabs, History and Exceptions
- 3) Third choice would be Jobs in a running state

- a. The bottom part will show two tabs, History and Exceptions

2.0 Architecture

The following screen shows you Create Licensing Job screen:

You can specify various fields as seen in the screen shot.

The screenshot shows a window titled "Create Licensing Job". The form contains the following fields and sections:

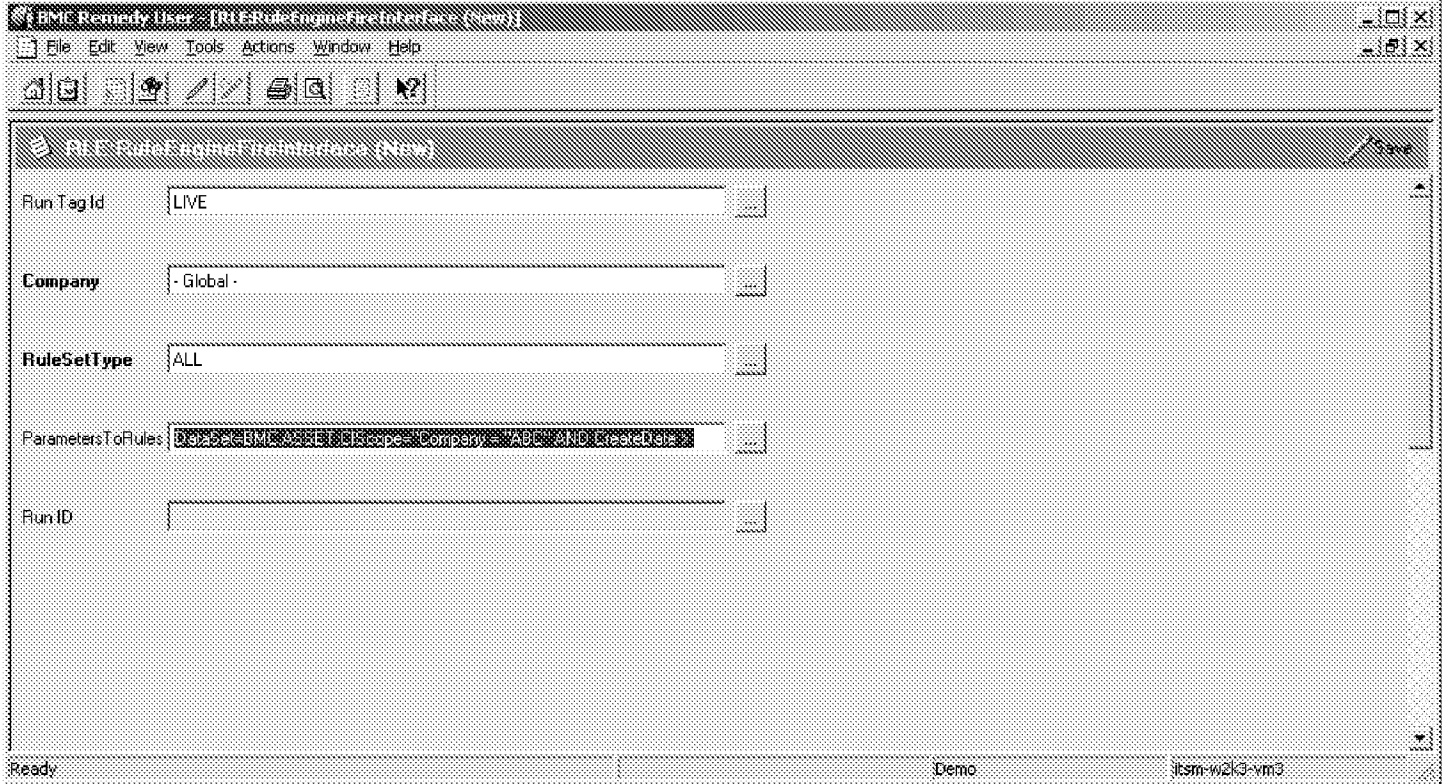
- Company***: Text input field.
- Job Name***: Text input field.
- Job Status***: Dropdown menu with "Active" selected.
- Type**: Text input field with a menu icon.
- Job Criteria**:
 - License Type**: Dropdown menu.
 - Manufacturer**: Text input field with a dropdown arrow.
 - Product Name+**: Text input field with a dropdown arrow.
 - Model/Version**: Text input field with a dropdown arrow.
 - Categorization Tier 1**: Text input field with a dropdown arrow.
 - Categorization Tier 2**: Text input field with a dropdown arrow.
 - Categorization Tier 3**: Text input field with a dropdown arrow.
- Advanced Certificate Criteria**: Text input field with a menu icon and an "Advanced Search" link.
- CI Dataset**: Dropdown menu.
- Advanced CI Criteria**: Text input field with a menu icon and an "Advanced Search" link.
- Created By***: Text input field containing "\$USER\$".
- Last Modified By**: Text input field.
- Create Date**: Text input field.
- Modified Date**: Text input field.

A "Close" button is located at the bottom left of the form. A "Palette" sidebar is visible on the right side of the window.

The Type menu will have two choices, ALL and COMPLIANCE. When the Job type is ALL, this will be used to run both Connection and Compliance, when type is COMPLIANCE it will only run it for Compliance.

Calling the engine:

When we want to call the engine, we will need to push an entry in the RLE:RuleEngineFireInterface form



RunTagId will be LIVE, Company would be set with the Company from AST:LicenseJob, RuleSetType would be from the Type field.

If Type Field ALL: This means run both connection and compliance. Means we have to query CI's to connect and then run compliance

For the ParametersToRules we will need to construct the string based on the values for various other fields on the AST:LicenseJob

The values for following parameters need to be constructed and then concatenated to push to the ParametersToRules field

1. DataSet
2. CIScope
3. CertificateScope

- 1) DataSet parameters will be based on the DataSetID field.
- 2) CIScope will be another parameter and the qual that will get built for it would be based on
 - a. Company
 - b. Product Cat stuct
 - c. Advanced CI Criteria.

The Advanced search will let you pick fields from base element only.

- 3) CertificateScope will be another parameter and the qual that will get built for it would be based on
 - a. Company
 - b. Product Cat stuct
 - c. License Type
 - d. Advanced Certificate Criteria

The DataSet, CIScope will get used when searching for CI's and CertificateScope will get used when searching for Certificates.

Note: As part of the connection/compliance run, we look up certificates currently based on License Type = "xxx" AND Can Connect = Yes.

We will need to append @CertificateScope@ parameter in the rules for querying certificates.

If Type Field COMPLIANCE: This means run only compliance. Means we don't have to query CI's to connect and only run compliance rules make sure everything is in compliance

1. When the job is defined for COMPLIANCE only, we will not need Dataset and CIScope parameters as they are not used for Compliance.
2. So when type is compliance we should disable fields that are not relevant, which is anything specific to CIScope, but if the fields are shared for Connection and Compliance then they would be enabled.

1.0 Solution, Product or Component Overview

1.1 Business Requirement

This document will define the business requirements for the new Contract Management.

1.2 Functionality Description

1.2.1 Business rules for a Permission Model (S1482)

- A viewer can see all un-restricted contracts (there will be a flag in the 'Contract' form – Restricted (the default value will be un-restricted)
 - Not Viewable Flag (Restricted)
 - Not Editable
- Create new User Role + being in a group assigned as a manager (a user could be a Manager) (only my group contracts)
- Software Asset Manager
- Will have a default 'Viewer' (contract viewer) access
- Will have a default 'Asset Viewer' access
- A software asset manager will be able to manage certificates if he also gets the 'User' role (contract user).
- Groups existing in the application today:
 - Contract Full Access -> Computed group: "Administrator" OR "Asset User" OR "Asset Admin" OR "SLM Manager" OR "SLM Config" OR "SLM Unrestricted Manager"
 - Contract Administrator -> Computed group: "Administrator" OR "Asset Admin" OR "SLM Manager" OR "SLM Config" OR "SLM Unrestricted Manager"
 - Contract View Only -> Computed group: "Administrator" OR "Asset Viewer" OR "SLM Manager" OR "SLM Config" OR "SLM Unrestricted Manager"
- Create Restricted/Unrestricted attribute on CTR:ContractBase and all AST:Contract forms
- If a contract is set to Restricted, this contract will be editable by Contract Full Access and Contract Administrator groups. Contract View Only will not be able to see the Restricted record.
- New roles will be created similar to the Purchasing roles:
 - Contract Administrator, Contract User and Contract Viewer
- New groups will be created similar to the Purchasing groups:
 - Contract Administrator, Contract User and Contract Viewer as a Regular groups

- Each of the new roles will be map to the new Group Permission for Contract Management Application and Asset Inventory Application (2 records created for each role)
- All new Contract permission Groups will be added to all Contract forms in permission
- In people form, user has to be defined with Contract group permission to access the contract deployable application
- New data created in LIC:SYS-License Permission Map form, need localization

1.2.2 Functionality Description for Generic Contract (S701).

Enable new contract types to be created via data vs. customization

- Needs to be able to manage new contract types easily
- Provide an interface to create new contract type based on CTR:Contract Generic form
- This interface will be accessible from the Application Administration Console, in Asset Management/Advanced Options
- This interface will allow user to create a new contract type, create all necessary attributes already existing in the CTR:Contract Generic form

1.2.2.1 Documentation for Generic Contract

A detailed documentation will be provided to the customer to help them when enhancing contract types

Various options will be offered:

- If no custom attributes on the CTR:Contract_Generic form, it will be possible to create a new view of this form to expose the new contract type
- If new attributes are created, a set of instructions will be provided to explain how the AST:AssetContract forms work, how to create this custom form, attach all necessary workflow, etc.

1.2.3 New Forms to be created

At Foundation level (This is loaded in foundation)

Form Name	Definition	Comments
CTR:Master Contract_	New form for Contract Management Similar to AST:ASsetSupport_ form	Add to Remedy contract management deployable application Need to be localized

Form Name	Definition	Comments
CTR:Master Contract	<p>New form for Contract Management</p> <p>Join form between CTR:ContractBase and CTR:Master Contract_</p> <p>Similar to AST:ASsetSupport form</p>	<p>Add to Remedy contract management deployable application</p> <p>Need to be localized</p>
CTR:Terms_Conditions	<p>New form for Contract Management</p>	<p>Add to Remedy contract management deployable application</p> <p>Need to be localized</p>
CTR:Rights_Granted	<p>New form for Contract Management</p>	<p>Add to Remedy contract management deployable application</p> <p>Need to be localized</p>
CTR:Contract_Relationship	<p>New form for Contract Management relationship</p>	<p>Add to Remedy contract management deployable application</p> <p>Regular form for Contract relationship between a Master contract and any other contract types and Contract relationship between contract and any other contract types and relationship between any contract type and Purchase order Line Item</p> <p>Need to be localized</p>
CTR:Contract_Relationship_Parent	<p>New form for Contract Management relationship localization</p>	<p>Join form for Contract relationship</p> <p>Add to Remedy contract management deployable application</p>
CTR:Contract_Relationship_Child	<p>New form for Contract Management relationship localization</p>	<p>Join form for Contract relationship</p> <p>Add to Remedy contract management deployable application</p>

Form Name	Definition	Comments
CTR:Contract_Relationship_Parent_Locale	New form for Contract Management relationship localization	Join form for Contract relationship for localization Add to Remedy contract management deployable application
CTR:Contract_Relationship_Child_Locale	New form for Contract Management relationship localization	Join form for Contract relationship for localization Add to Remedy contract management deployable application
CTR:Contract_Generic_	New form for Contract Management to give the ability to create new contract types	Need to be localized Add to Remedy contract management deployable application
CTR:Generic Contract	New form for Contract Management to give the ability to create new contract types Join form CTR:ContractBase CTR_Contract_Generic_	Add to Remedy contract management deployable application
CTR:CreateContractType	New form for Contract Management to give the ability to create new contract types	Need to be localized Add to Remedy contract management deployable application
New record will need to be created in CFG:HTMLCatalog for new Contract Type to display the detailed information	New for Contract Management to give the ability to create new contract types	These records will have to be added manually by customer and instructions will be provided. Each of these records will need to be localized
CTR:Worklog	New form to hold the Work Info for each contract type	
AST:Contract_Contact_Dlg	Dialog form to update the Contact information for each Contract Type	

AT ASSET Level

Form Name	Definition	Comments
AST:License Certificates		AST:Keys and versions already exist and very similar to what we want. We can rename the existing form and add the missing attributes Need to be localized
AST:LicenseCertificatesSearch		Dialog form to search for existing License Certificate to relate to a Software License Contract, see Section 1.3.7 Add to Remedy Asset Inventory deployable application Need to be localized

1.2.4 Definition of attributes Create CTR:Contract_Generic_ and CTR:Contract_Generic form

Note: If we want to have all the AST Contract functionality, we need to add the Contact tab, Work Info tab, Child Contract tab, Payments tab, Relationship tab to the CTR:Contract_Generic form and create a Default Admin view and a dialog view. And attach all the existing workflow to the new AST contract form using driver.exe

- Definition of attributes for CTR:Contract_Generic_
 - CTR:Contract_Generic_ form will be a copy of CTR:MasterContract_ form
 - All attributes from CTR:MasterContract_ will be inherited into CTR:Contract_Generic_ form
 - All workflow from CTR:MasterContract_ will be inherited into CTR:Contract_Generic_ form
- Definition of attributes for CTR:Contract_Generic
 - CTR:Contract_Generic form will be a copy CTR:MasterContract form
 - All attributes from CTR:MasterContract will be inherited into CTR:Contract_Generic form
 - All workflow from CTR:MasterContract will be inherited into CTR:Contract_Generic form
 - CTR:Contract_Generic will be used to view/manage new contract type

Note: See Section 1.2.5 for a list of all attributes

1.2.4.1 New Record to be created in RAC forms for Generic Contract

Not necessary if we can use the Contract console to create new contract types

From RAC:Config:Application form, a new application will be added for Contract Management

From RAC:Config:Options form, a new option will be added for Advanced Options

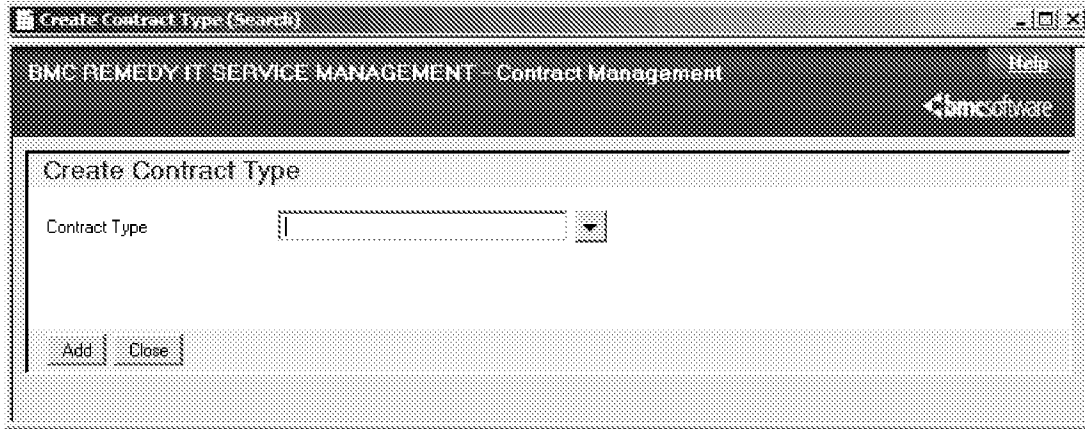
From RAC:Config:Task form, a new task will be added for Create new contract type

The new record created will need to be localized

Form Name		Comments
RAC:ConfigOptions	Add Task	Create Contract Type
	Description for the new record	
Option Name	ASTAdvanced	
Task Name	Create New Contract Type	
Console Text	Create New Contract Type	
Task Type	Standard	
Description	Create New Contract Type	
Accessible Groups	Asset Config	
Display On Console	Yes	
Status	Enabled	
Data Tags	Config-AST	
Config Form	AST:ConfigContractType	
WUT View Name	Default User View	
Web View Name	Default User View	
Form Open Mode	Search	

1.2.4.2 Definition of attributes for CTR:CreateContractType form for Generic Contract

This new form will be similar to AST:ConfigOutage form and will give the functionality to create a new contract Type



Attribute Name (Label)	DB Name	Comments
Contract Type		This value will be used in the Menu to display the new contract type Example: Statement of Work
Contract Key Value		System Generated
Contract Menu Order		System Generated
Add		Create new Contract Type
Close		Close form

1.2.5 Definition of attributes for Base Contract

Attribute Name (Label)	DB Name	Comments
Contract ID	Contract ID+	Field ID = 260000000
Contract Type		Field ID = 300580400
Status	Status	Field ID = 7
Status Reason	Status Reason	Field ID = 1000000150
Company	Company	Field ID = 1000000001
Organization	Organization	Field ID = 1000000010

Attribute Name (Label)	DB Name	Comments
Department	Department	Field ID = 100000011
Supplier	Supplier Name+	Field ID = 1000000396
Supplier Organization	Supplier Organization	Field ID = 1000003662
Supplier Group	Supplier Group	Field ID = 1000003663
Supplier Phone	Supplier Phone	Field ID = 240001011
Supplier Hotline	Supplier Hotline	Field ID = 240001012
Authorized Callers	Authorized Callers	Field ID = 260100021
Type (change to Term)	Type	Field ID = 200000103
Notification Date	Notification Date	Field ID = 240001015 Modify the field attribute to be Optional
Start Date	Start Date	Field ID = 260000001
Termination Date	Expiration Date	Field ID = 240001000 Modify the field attribute to be Optional
Purchase Date	Purchase Date	Field ID = 260000006
Cost Center	Cost Center	Field ID = 260100006
Budget code	Budget code	Field ID = 260100007
Project Number	Project Number	Field ID = 260100005
Accounting code	Accounting code	Field ID = 240000004
Purchase cost	Purchase cost	Field ID = 260600003
Late Charge	Late Charge	New field to be created
Net Days payable	Net Days payable	New field to be created
Tax	Tax	New field to be created
Annual fee	Annual fee	New field to be created
Termination conditions		New field to be created

Attribute Name (Label)	DB Name	Comments
		From here, all attributes already exist in CTR:Base Contract form and need to exist in this new version
Support Company*	Assigned Support Company	Field ID = 1000000251
Support Organization*	Assigned Support Organization	Field ID = 1000000014
Notification Group+	Notification Group+	Field ID = 260142102
Notification Contact+	Notification Contact+	Field ID = 240001013
Cost per Asset	Cost per Asset	Field ID = 270002042
Renewal Cost	Renewal Cost	Field ID = 260600004
Cost per Component	Cost per Component	Field ID = 270002043
Parent Contract ID	Parent Contract ID	Field ID = 300441600
InstanceID	InstanceID	
Notification to Review (date field)	Notification_Review	New field to be created Field ID = Used for Rolling Software Contract See section 1.3.1
Expiration Date Selection	Expiration Date Selection	New field to be created Selection field See section 1.3.1.1 for business rules definition
Ownership Group	ContractOwnerGroup	New field to be created Field ID = 303515200
Ownership Contact	ContractOwnerContact	New field to be created Field ID = 303515300

Attribute Name (Label)	DB Name	Comments
Work Info Type	z1D_Activity_Type	1 generic Work Info Type will be created to manage Contract work log
Restricted/Unrestricted		See permission model This field will need to be created in all AST contract forms

1.2.6 Definition of attributes for CTR:Master Contract_

This form will be similar to AST:AssetSupport_

No other specific attributes are required for CTR:Master Contract_

1.2.7 Definition of attributes for CTR:Master Contract

This form will be similar to AST:AssetSupport_

Join form between CTR:ContractBase and CTR:Master Contract_

All attributes from CTR:ContractBase will be inherited into CTR:Master Contract

The list of attributes below will be created to provide the ability to manage Master Contract properly

Attribute Name (Label)	Definition	Comments
z2TH_Contract	Table to show all related contract in a tree view Similar to the functionality in Computer system Relationship Details	Table field Tree View Field ID = 303505400
z2TH_ContractDetails	Table to show all related contract individually Similar to the functionality in Computer system Relationship Details	Table field Detail View Field ID = 303505800

Attribute Name (Label)	Definition	Comments
ContractRelation_Details_View	New records will need to be created in CFG:HTMLCatalog for each Contract Type to display the detailed information Each of these records will need to be localized	View Field Field ID = 303517300
z2TH_MC_Cost	Table to show all related cost for the corresponding record and all related contract	Table field
z2NI_Terms_and_Conditions	Menu to access Term and Conditions	Field ID = 303517600
z2NI_RightsGranted	Menu to access Rights Granted	Field ID = 303517500
Ownership Group	Attribute to select the owner group from the Support group form	Selection field
Ownership Contact	Attribute to select the owner person from the People form	Selection field
z2NI_Contract_Contact	Menu to access Contract This replace the Contact tab	Field ID = 303517700

1.2.7.1 SHR:SchemaNames Record for Master Contract

Create new record in SHR:SchemaNames form to manage the Master Contract

Attribute Name (Label)	Value	Comments
ClassID01	CTR:MasterContract	
ClassID02	CTR:MasterContract	
Description	CTR:MasterContract	
Form Code	CMC	
Has Asset UI	No	
Lookup Keyword	ASSTEMASTERCONTRACT	
Proper Name	Master Contract	

Attribute Name (Label)	Value	Comments
Purchase Requisition	No	
Schema Name	CTR:MasterContract	
Show In Menu	Yes	
Status	Current	
User Class Name01	Master Contract	
User Class Name02	Master Contract	

1.2.8 Definition of attributes for CTR:Terms_Conditions

This form will be created with the same model as AST:WorkLog form and similar functionality

Attribute Name (Label)	DB Name	Comments
Contract InstanceID	Contract InstanceID	Hold the Instance ID for the related contract type (like Software contract)
Parent Contract InstanceID	Parent Contract InstanceID	Hold the Instance ID for the Master Contract
Description	Description	
Attachment	Attachment	
InstanceID	InstanceID	

1.2.9 Definition of attributes for CTR:Rights_Granted

This form will be created with the same model as AST:WorkLog form and similar functionality

Attribute Name (Label)	DB Name	Comments
Contract InstanceID	Contract InstanceID	
Description	Description	
Attachment	Attachment	
InstanceID	InstanceID	

1.2.10 Definition of attributes for CTR:WorkLog

This form will be created with the same model as AST:WorkLog form and similar functionality

Attribute Name (Label)	DB Name	Comments
Contract InstanceID	Contract InstanceID	
Description	Description	
Attachment	Attachment	
InstanceID	InstanceID	

1.2.11 Definition of attributes for CTR:Contract_Relationship

This form will be similar to AST:CMDB Associations and will hold all relationship between Master contract and all other contract types, relationship between all contract types and relationships between contract and Purchase Line Item

VUI ID to be defined later

Attribute Name (Label)	DB Name	Comments
Parent Form Name		303503800
Child Form Name		303504400
Parent Instance ID		303503900
Child Instance ID		303504500
Parent Relationship Type		303504000
Child Relationship Type		303504600
Parent Description		303504100
Child Description		303504700
Parent Status		303504200
Child Status		303504800
Parent LookupKeyword		303504300
Child LookupKeyword		303504900
Association Type		1000000208

Attribute Name (Label)	DB Name	Comments
Relationship ID		Field ID = 1
Status		Field ID = 7

1.2.12 Definition of attributes for CTR:Contract_Relationship_Parent

Join form	Join Criteria	Comments
CTR:ContractRelationship	Parent Instance ID	
CTR:ContractBase	InstanceID	
		All attributes from CTR:ContractBase will be present on this form and only necessary attributes from

1.2.13 Definition of attributes for CTR:Contract_Relationship_Child

Join form	Join Criteria	Comments
CTR:ContractRelationship	Child Instance ID	
CTR:ContractBase	InstanceID	
		All attributes from CTR:ContractBase will be present on this form and only necessary attributes from

1.2.14 Definition of attributes for CTR:Contract_Relationship_Parent_Locale

Join form	Join Criteria	Comments
CTR:Contract_Relationship_Parent	Parent LookupKeyword	
SHR:SchemaNames	Lookup Keyword	

Join form	Join Criteria	Comments
		All attributes from CTR:Contract_Relationship_Parent will be present on this form and only necessary attributes from and only necessary attributes from SHR:SchemaNames to support localization like Proper Name

1.2.15 Definition of attributes for CTR:Contract_Relationship_Child_Locale

Join form	Join Criteria	Comments
CTR:Contract_Relationship_Child	Child LookupKeyword	
SHR:SchemaNames	Lookup Keyword	
		All attributes from CTR:Contract_Relationship_Child will be present on this form and only necessary attributes from and only necessary attributes from SHR:SchemaNames to support localization like Proper Name

1.3 Functionality Use Case

1.3.1 Use Case Diagram

N/A

2.0 Architecture

3.0 Solution, Product or Component Design

3.1 Diagrams

3.2 User Interface Design

N/A

3.3 Workflow Pseudo Code

This section describes the workflow to create/modify to implement the new contract management business requirements

3.3.1 All Use Cases

3.3.1.1 Business rules for New Contract Type:

In CTR:ContractBase form we are creating a new attribute for Contract Type

This attribute will be populated during submit and modify

Some functionality will be created for upgrade using the attribute zTmpKeyword to update properly the Contract Type menu selection

Filter Name		Comments
CTR:SHR-Set-ContractType_Update		

3.3.1.2 Business Rules for Contract Terms (S1227)

- Modify Contract terms for software contract to
 - Never ending
 - Rolling contract (end date is changeable) what is the expectation for the date change (rules based?)
 - Fixed
- Need to create/modify SYS:Menu Items data
- For now Term attribute will be modified only for Software contract. The other contract types will remain with their actual definition
- We will create an Expiration Date Selection to have a smaller increment (30 days, 60 days, 90 days in the case of a temp contract or a services contract)
- The Expiry date will be calculated automatically according to the Expiration Date Selection
- On CTR:Contract Base form the Expiration Date and Notification Date will need to be modified to set it "Optional"
- Workflow will need to be created for each Contract Type (Support, Lease, etc) to handle the Expiration Date and Notification Date as "Required" fields
- Now, the Expiry date is a date field, and user needs to select a date from the calendar. We will add some functionality to have a selection by days/months/year and calculate the date accordingly
- For Never Ending contract, the expiry date and notification date will be disabled

- For Rolling contract, change the label for Notification Date to Notification to Review
- Build business rules for rolling contract
- For Fixed contract, Expiration date and Notification date will become a required attribute via workflow
- For Master Contract this new Term will be implemented.
- For Master Contract no specific business rules are defined to handle the term except the ones define for Software contract
- For Master Contract, with Related sub-contract, Term will be independent form sub-contracts term and the expiration dates on Master contract and all their sub-contracts are manage independently
-

3.3.1.3 Status Attribute Modifications (S1225):

Definition for the actual and proposed values for Status attributes on CTR:Base Contract form, with new proposed values and recommendation for upgrade installation

- Actual Values for Status attributes:
 - Current, Pending Renewal, Expired
- Proposed New Values for Status attributes:
 - Draft, Executed, Historical, Delete
- Business rules for Status Transitions
 - Draft to Executed: If a contract start date is reach, Status = Executed and Status Reason = Active
 - Draft to Historical: If a contract was never signed, should be move to Historical with Status Reason = Cancelled
 - Draft to Delete
 - Executed to Historical: If expiry date is reach, Status = Historical and Status Reason = Expired
 - **Can we ‘renew’ contract after it is ‘expired’, to check with Tom, the chart below says the Status Reason can go to In Effect after Expired or Terminated, In Effect was never defined.**
 - Historical to Draft, only if Status Reason = Cancelled
 - Delete: If Status Reason = Scheduled for Deletion, Status will be moved to “Delete”

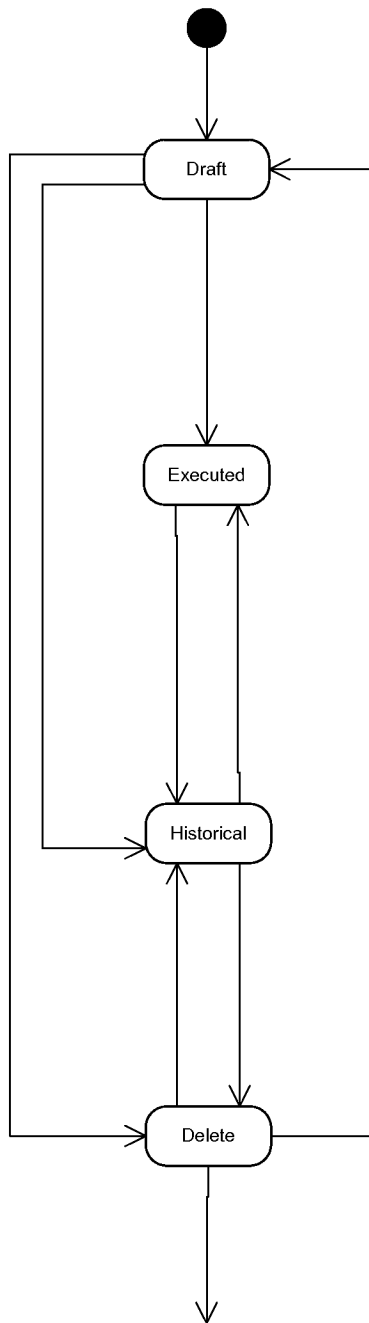
3.3.1.4 Status Reasons Attribute Modifications (S1225):

Definition for the actual and proposed values for Status Reasons attributes on CTR:Base Contract form, with new proposed values and recommendation for upgrade installation

New records created in SYS:Status Reason Menu Item form where Form Name = CTR:ContractBase

These records need to be localized

- Actual Values for Status Reasons attributes:
 - Gets the same values as Status, Current, Pending Renewal, Expired
- Proposed Values for Status Reason attributes:
 - For Draft: In negotiation, Pending Signature:
 - For Executed: Active, Requires Attention, Under re-negotiation, Change Pending, On Hold
 - For Historical: Expired, Terminated, Cancelled
 - For Delete: Scheduled for Deletion
- Business rules for Status Reason Transitions:
 - In Negotiation: Still under negotiation
 - Pending Signature: Negotiations and review completed but not signed
 - Active: Default Value
 - Requires Attention: No activity on the contract for some time (configurable)
 - Under re-negotiation: Possible Change Pending
 - Change Pending: Change on the contract is pending
 - On Hold: No possible activity
 - Expired: Reach Expiry Date
 - Terminated: Contract was terminated at Expiry date
 - Cancelled: Contract was terminated before Expiry Date
 - Scheduled for Deletion: Contract can be removed



Draft – Contract has never been executed but is going through the process of being executed

Valid Status Reasons:

- In Negotiation – Still negotiating contract
- Pending Signature – Negotiations and review completed but not signed
- Other?

Valid transitions:

- Executed – once executed, status in-effect
- Historical – if the contract was never signed, should be saved as historical with reason of never executed
- Delete – option to move contract right to Delete if it was never Active

Executed – Contract is executed and valid

Valid Status Reasons:

- Active – default status reason (all ok)
- Requires attention (?) – still valid but there has not been activity on the contract for some time (how much – config)
- Under re-negotiation – There is a possible change pending on the contract but still is valid
- Change Pending – there is a change to the contract that is pending – could be an early termination, addendum, amendment, etc...
- On Hold – still valid but no transactions should be performed against it
- Other?

Valid transitions:

- Historical – At any time, expiry date should be completed. Status reason is ?
- ENGINE IS CALLED TO RECALCULATE COMPLIANCE RULES

Historical – Contract is not valid

Valid Status Reasons:

- Expired – Contract has expired
- Terminated – Contract was terminated when effective
- Cancelled – Contract was cancelled before effective
- Other?

Valid transitions:

- In Effect – Only when status reason is expired or terminated, default status is ?
- Draft – Only when the status reason is Cancelled
- ENGINE IS CALLED TO RECALCULATE COMPLIANCE RULES

Delete – Contract is Scheduled for deletion

Valid Status Reasons:

- Scheduled for Deletion – Contract is going to be removed on the next delete run

Valid transitions:

- Back to previous state – either Historical or Draft

3.3.1.5 Business rules for Terms and Conditions (S1573)

- A new link in the Navigation bar will be created on Software contract for Terms and conditions and Rights Granted information
- User will be able to View/Create/Remove related Terms and conditions and Rights Granted
- When a new Term and conditions is created and a sub-contract (for example Software License contract) is related, the Term and Conditions will also be copied to the sub-contract if no Term and Conditions are present for the sub-contract.

- If the Term and conditions for the sub-contract has been modified, the Term and Conditions for the Master Contract will not be copied into the sub-contract.

3.3.2 Definition of workflow for Creating new Contract Type (Generic Contract S701)

3.3.2.1 Creating Generic Contract From the Application Administration Console/Asset Management/Advanced Options

Attribute Name (Label)	Details	Comments
Add new Contract Type	From the Application Administration Console, in Asset Management/Advanced Options	Open CTR_CreateContractType
Add	Create new contract type	Push value to SYS:Menu items
Add	User created a new Contract Type	Puch Value to SHR:SchemaName fomr
Close		Close form

3.3.3 Business rules for a Master Contract (S1226)

- **All workflow from AST:AssetSupport will be attach to CTR:MasterContract. This workflow is not part of the Contract Management Deployable application but to the Asset management deployable application and will have AST: in the naming convention for Application**
- A Master Contract will have the ability to be related to any other Contract types

BMC REMEDY IT SERVICE MANAGEMENT - Contract Management

Master Contract

ID*+ : MCD01

Master Contract Information

Summary* MCD01 **Company*+** My Company **Region** **Term*** Fixed **Organization** **Site Group** **Status*** Draft **Department** **Site+** **Supplier Name*** Supplier **Status Reason** [Profile](#)

Accounting **Cost Center*** Unallocated **Purchasing** **Start Date** **Purchase Date** **Budget Code** **Purchase Cost** 0.00 USD **Project Number** **Accounting Code** **Contract Supported By:** **Support Company*** My Company **Expiration Date*** 3/31/2008 12:00:00 AM **Support Organization*** IT Support Organization **Notification Date*** 3/23/2008 12:00:00 AM **Notification Group*+** Internal Support **Owner Group** **Notification Contact** **Owner Contact**

[General](#) [Work Info](#) [Related Contracts](#) [Payments](#) [Relationships](#)

- A new tab will be created to show the list of related contracts
- Contract relationship between master contract and any other contract types should be build on the same model of the relationship details used for computer system in patch 007. 2 tables, one as Tree View to show Contract Type and one List View to show the contract details. (See picture below)

[General](#) [Work Info](#) [Related Contracts](#) [Payments](#) [Relationships](#)

Related Contracts

[-] Software License
 [-] LIC001
 [-] LIC002
 [+] Support

Contract ID+	Expiration Date	Notification Date
LIC001	4/30/2008 12:00:00 AM	4/24/2008 12:00:00 AM
LIC002	3/31/2009 12:00:00 AM	3/31/2010 12:00:00 AM

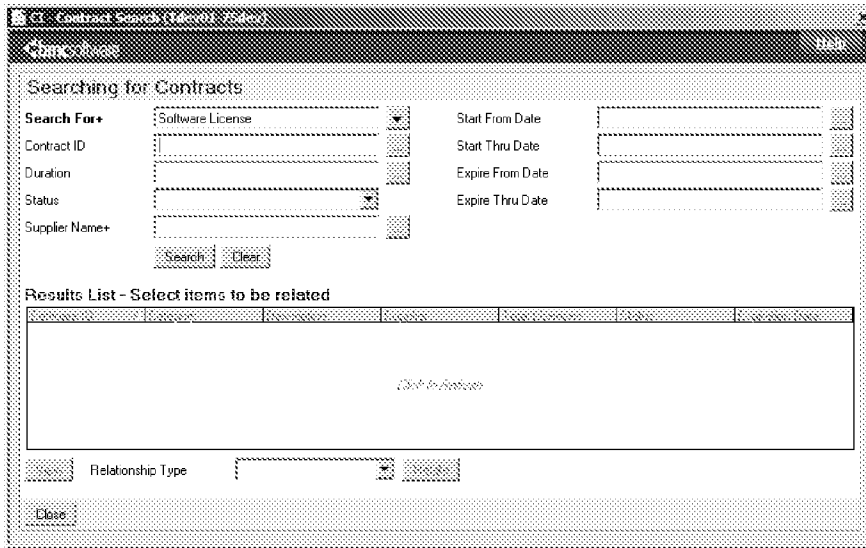
[View](#) [Remove](#)

Create New Contracts **Search Existing Contracts**

Contract Type [Create](#) Contract Type [Search](#)

- Create all necessary join forms to handle the contract relationship and localization:
- When a relationship is created between contract, all necessary information will be push to the new CTR:Contract Relationship form for the parent and for the child record
 - Parent Form Name, Parent Instance ID, Parent Status, Parent Contract ID, Parent Contract Type,
 - Child Form Name, Child Instance ID, Child Status, Child Contract ID, Child Contract Type,

- User will be able to relate an existing contract to a master contract using the dialog SHRAAS:SHR:SearchPanel or from the Contract Console



- User will be able to create a new contract of any type and related to a master contract (See picture above)
- A contract type created and related from a Master Contract will inherit some attributes define for the Master contract.
 - Parent Contract ID, Company, Department, Organization, Customer ID, Supplier, Cost Center, Support Company, Support Organization, Notification Group, Notification Contact, Owner Group, Owner Contact,
- User will be able to remove a contract related to a master contract (Remove button on the picture above)
- User will be able to view a related contract (View button on the picture above) or Double-click in the table
- All other functionality of a contract will be available to a master contract like Accounting, purchasing, Contract support by, dates, contacts, payments, work info, etc.
- From a related contract user will be able to view the master contract using a View Master Contract button.
- If the status of a Master Contract is change, no modification will be made to the status of the related contracts
- When the expiration date is reach on a Master Contract, the status will be modified according the Status business rules
- When the Notification date is reach on a Master Contract, the notification will be send to the Notification Group or Notification contact
- Role up costs from all my sub contracts (contained contracts) the Role up cost field will be updated
- Cost related to a child contract will be available to View only, it will not be possible to remove and/or add these costs

- Add or removing costs from a Master Contract will be possible only for cost related to this master contract
- **Will we have a cost for a master contract, or is always a roll up of costs.**
- Term and Conditions information will be available from the Term and Conditions link in the function menu
- This link will be available for all contract types
- Add or removing Term and Conditions from a Master Contract will be possible only for Term and Conditions related to this master contract
- In the General tab, a new set of attributes and functionality will be added for the ownership contract information

Attribute Name (Label)	DB Name	Comments
Ownership Group		Query the support Group form
Ownership Contact		Query the support People form

- These attributes will be created in all other contract type forms
- If Owner Group is different form the Notification Group, a notification will be send to the Owner Group
- If Owner Contact is different form the Notification Contact, a notification will be send to the Owner Contact
- New Attributes will be added to manage Region/Site Group and Site for each contract
- These attributes will be added to all contract types
- A new dialog will be created to view and manage Certificates information for related contracts
- This dialog will be available in the Software contract form and the Master Contract form
- Add or removing Certificates from a Master Contract will be possible only for Certificates related to this master contract
- Rights Granted information will be available on a link in the function menu for related contracts
- This link will also be created in all other contract type forms
- Add or removing Rights and Restrictions from a Master Contract will be possible only for Rights and Restrictions related to this master contract

3.4 Permission Model

For Generic Contract, Asset Config user or Contract Administrator will have access to the Create New Contract Type option.

3.5 Upgrade Requirements

3.5.1 Business Rules for Contract Terms upgrade (S1227)

- Need to create SYS:Menu Items data
- Existing data in SYS:Menu Items form will be disabled
- All existing Software Contract will have the “Fixed” value after upgrade

3.5.2 Business rules for Status upgrade (S1225):

- All existing contract with a Status = Pending Renewal will be change to Executed and a corresponding Status Reason, see below.
- All existing contract with a Status = Current will be change to Executed and a corresponding Status Reason, see below.
- All existing contract with a Status = Expired will change to Historical and a corresponding Status Reason, see below.
- All existing contract will not have a Status has Delete
- Some temporary fields will need to be created to move the Status field old values to the new Status value

Filter Name		Comments
CTR:SHR-Set-Status_Upgrade_001		On modify All Action, set z1d_Action field to ("UPGRADESTATUS")"
CTR:SHR-Set-Status_Upgrade_002		
CTR:SHR-Set-Status_Upgrade_003		

3.5.3 Business rules for Status Reason upgrade (S1225):

- All existing contract with a Status = Pending Renewal will be change to Status = Executed and Status Reason = Change Pending
- All existing contract with a Status = Current will be change to Status = Executed and Status Reason = Active
- All existing contract with a Status = Expired will be change to Status = Historical and Status Reason = Expired

- Some temporary fields will need to be created to move the Status field old values to the new Status value

3.5.4 Business rules for New Contract Type upgrade :

In CTR:ContractBase form we are creating a new attribute for Contract Type

This attribute will have to be populated during upgrade

Some functionality will be created for upgrade using the attribute zTmpKeyword to update properly the Contract Type menu selection

Filter Name		Comments
CTR:SHR-Set-ContractType_Upgrade		On modify All Action, set z1d_Action field to "UPGRADECONTRACTTYPE"

3.5.5 Business rules for New Roles upgrade :

For existing user with Asset permission, we will need to add the new contract permission, example a asset user will have contract user.

Or we will document the new permission model and ask the customers to update their own user profiles for each user they will want to access the contract module. We think this is the better way since we cannot assume all existing asset users will have the same type of permissions for contract.

1.0 Solution, Product or Component Overview

The Advanced model interface provides an alternate way of setting questions for Connection and Compliance.

These would be typically used where you would need to do some complex operations to derive some data.

This provides more flexibility for the user to manipulate the questions data and hence map those manipulated data fields.

2.0 Architecture

The architecture mainly includes forms for capturing questions for both Connection and Compliance.

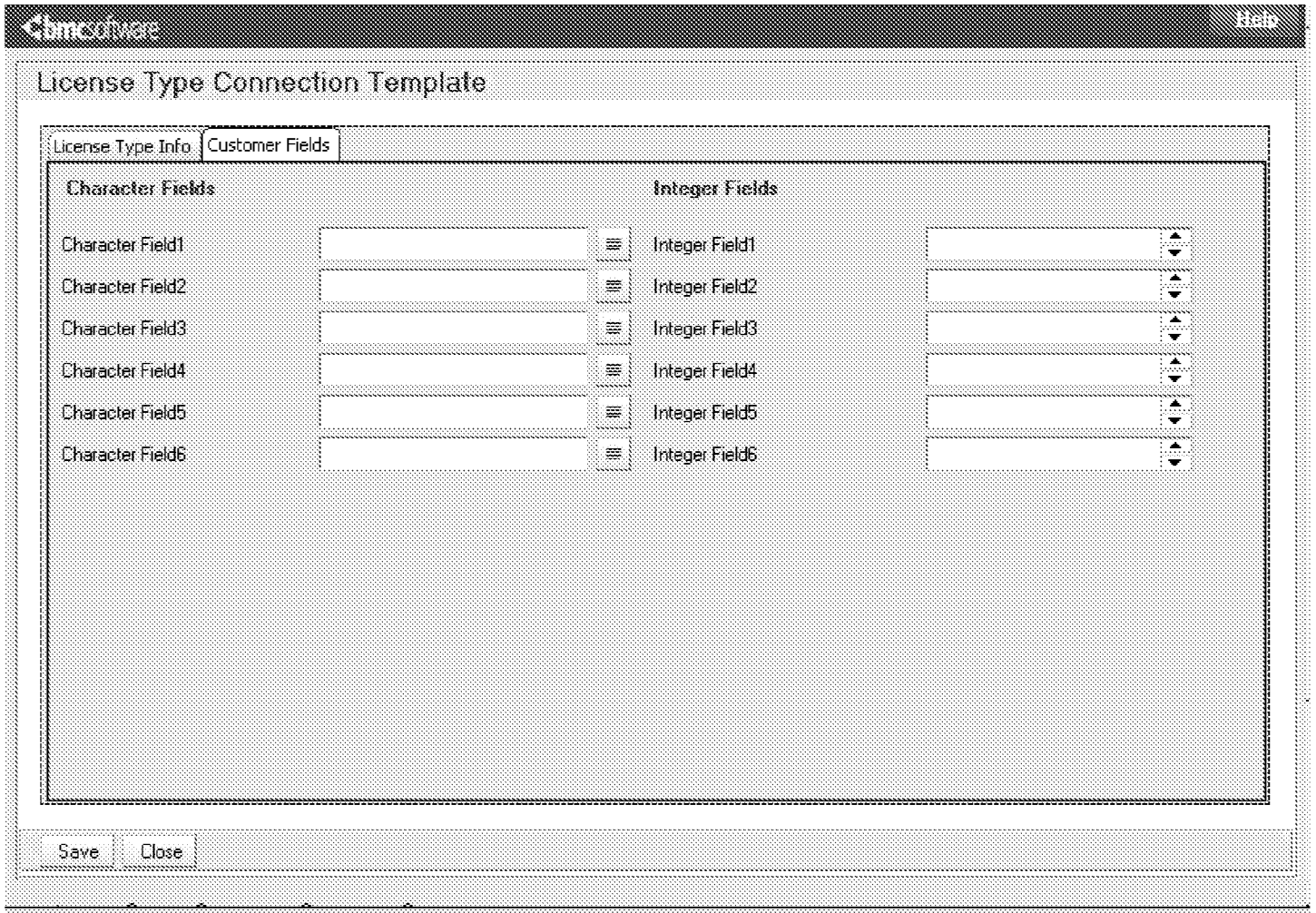
2.1 Template Form for Connection

A sample connection template regular form(AST:ConnectionTemplate_Advanced) has been created(see Screen1), it includes a set of fields that act as a anchor, which ties it back to the Certificate. These are parts of the Template and any advanced form will have those. These would be considered system fields e.g., License GUID, Company,Certificate ID, Certificate_InstanceID, MasterCertificateInstanceID,Certificate Group Status etc..

We also have display only fields License Name on this form. It will be set on open based on the License GUID.

Then we have the main body of the form and we have a customer fields area that contain optional fields in a certain field ID range, (303000500 to 303000900).Customers can use the template form as a sample form to change the field names, to add or delete fields and change the form name they want.

The Advanced forms have a Save and Close Button, so that when you open them from the Certificate, you can Save the answer to the questions and close the form.



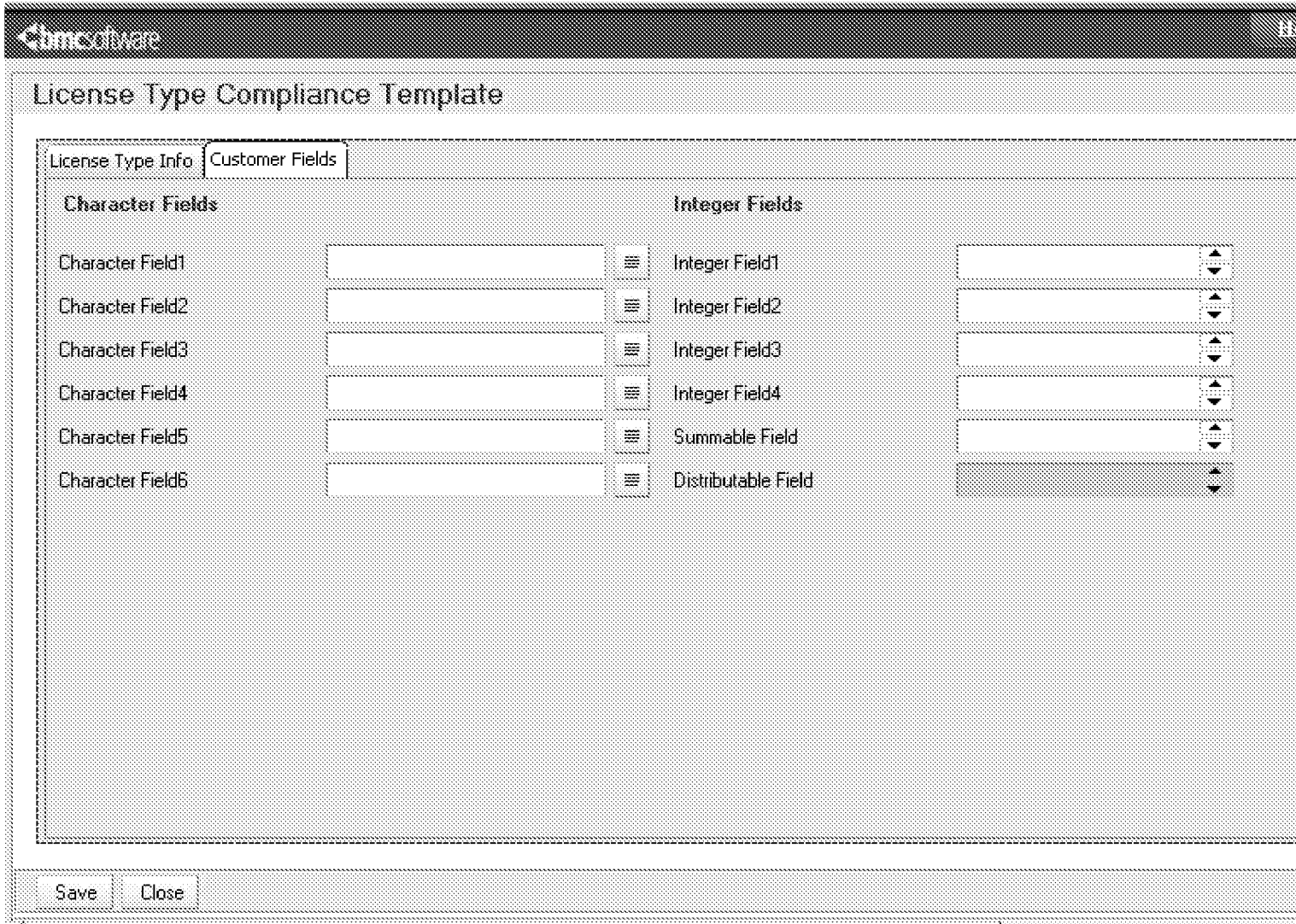
(Screen 1)

2.2 Template Form for Compliance

This follows the same guidelines as the Connection form.

A sample connection template regular form (AST:ComplianceTemplate_Advanced) has been created (Screen 2). Two integer fields Summable Field and Distributable field are part of the template. Summable Field behaves like # of purchased. Distributable Field behaves like # of Deployed. The Distributable Field will be tied to the Summable Field that we have on the template and we have workflow to distribute the count between the Children on this set of Distributable/Summable pair fields when working on group certificates.

If there is no grouping involved, the two fields will behave like two regular integer fields.



(Screen 2)

2.3 Integrating it with the Rule wizard

From the wizard you have the option to define if License Type is a Basic Model or an Advanced Model. If it is a Advanced model, there are workflow created for advanced wizard

For Connection

When user clicks the Next button on the License Type Tab, the wizard skips the questions tab and the user comes to the Mapping tab.

Instead of showing the list of questions to map, a SQL Menu has been created on the Mapping tab that shows all the fields from the connection template form (ex.AST:ConnectionTemplate_Advanced) within the above field id range. User can map the fields here and the mapping will be pushed to the AST:ConfigConnection_RuleFieldMapping.

The DB Names shown in the menu are always in English as this is a SQL menu. This SQL menu is dynamically created based on the runtime template form name customer input. No dependence on form names or specific fields.

For Compliance

Again for Compliance, the questions screen will be skipped; the user would go directly to the Actions step.

When clicking the next button on Mapping tab, following existing rule set workflow for compliance, there are separate workflow created for advanced model to push value to AST:ConfigCompliance_RuleReference and AST:ConfigCompliance_RuleDefinition.

Instead of looping the compliance question table to get all the compliance questions, create a view form named AST:Field View form. It is the view of the field table. Created a table on AST:LicenseTypeConfiguration to get all the customer field names within the ID range above on Compliance template form. This design is to make sure GetQuestions list on Rule menu of the Action form is dynamic. No Dependency on specific Compliance template form name or fields on the form.

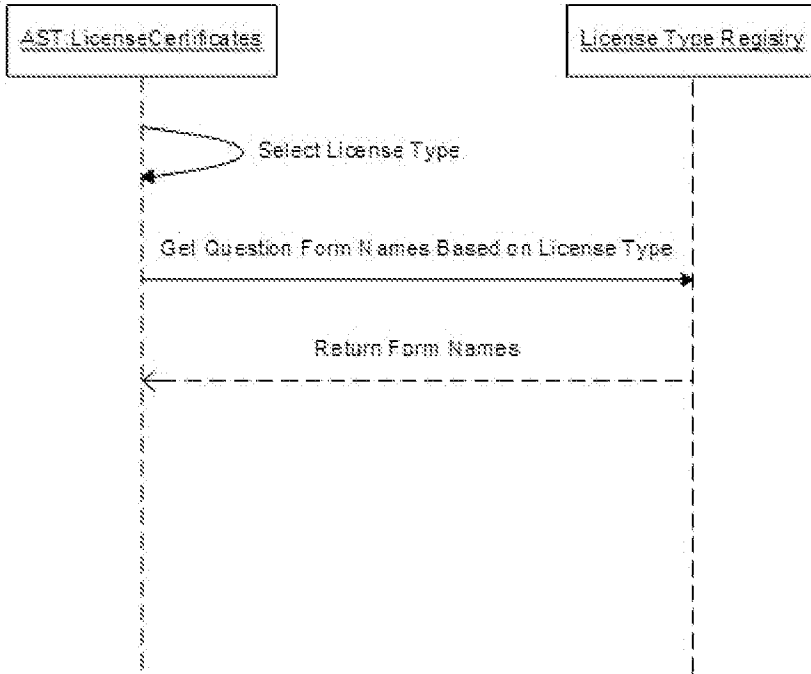
In the actions the user is able to select the compliance form for getting/setting data etc, basically the form is available to all the actions and it is just another AR form.

2.4 From the Certificate

From the Certificate, if the License Type is chosen which uses an Advanced Model, on going to the Connection Question step, it will open the Advanced Form for Connection..

Same for Compliance questions, when you are going into the Compliance question Setup, it will open the advanced form for Compliance for that specific license type.

When a user selects a license type, a call is made to the AST:LicenseTypeRegistry form to get the names of the question forms to open.



Advanced Model workflow on the Certificate Advanced Wizard is created .

Grouping certificates workflow on Advanced model has been implemented .Summable/Distributable function on grouping certificates is provided.

class that includes the map of AR Values by attribute names
we might get AR Values from CMDBInstance and AR Entry
class **CommonBO**

```
{
    Map<Attr name, Value> data;
    AR form name or CMDB table name
    String sourceName;
}
```

Global variables definition

```
List listRules;
Boolean endFlag;
Int nextPosAfterRule;
map that should save a CommonBO object accordingly to rule name
Map<Rule name, CommonBO > ruleResultsData;
```

Start the algorithm

```
void runRules(String ruleSetId)
{
    create map that should save a CommonBO object accordingly to rule name
    Map<Rule name, CommonBO> ruleResultsData;

    Get all rules by ruleSetId and save it as global variable
    List listRules = getRulesByRuleSetId(String ruleSetId);

    save ignore rules as global param
    List ignoreRules = getIgnoreRules(listRules);

    get the first rule from the list of rules
    processRule(0);
}
```

```
List<Rule> getIgnoreRules()
{
    call Remedy API to get ignore Rules
}
```

```
List<RuleSet> getRuleSets()
{
    call Remedy API to get Rule Sets
}
```

```
List<Rule> getRules(String formName, String ruleSetId)
{
    call Remedy API to get Rules from specific form by a rule set id
    convert a data from form to JAVA Rule class.
}
```

```
List<Rule> getRulesByRuleSetId(String ruleSetId)
{
    populate list of rules from all forms by rule type
    List listRules = getRules("GetInterfaceRulesForm", "SetId");
    listRules.add(getRules("LoopInterfaceRulesForm", "SetId"));
    .....

    sort list of rules by the rule's sequence
    listRules = sortRulesBySequence(listRules);

    return listRules;
}
```

Int processRule(int curRulePosition)

```
{
    Iterator = listRules.get(curRulePosition);

    If(iterator.hasNext()){
        rule = iterator.next();

        work by a rule type
        switch(rule.type)
        {
            case:START LOOP
            {
                int nextStartPos = iter.nextIndex();
                nextPosAfterRule = processRule(nextStartPos);
                if (endFlag.get()) {
                    endFlag.set(false);
                    nextPosAfterRule = processRule(nextPosAfterRule);
                } else {

```

```

        Throw Exception("Error - START rule without END");
    }
}
case:GET
{
    get results for specific rule
    List<CommonBO> results = doGet();

    save position of current rule
    Rule nextRule = list.next;

    for each member from results
    {
        replace an entry accordingly to rule name in the Map ruleResultsData
        ruleResultsData.replace(rule.name, result);

        enter to a recursion with next rule from the list
        int nextGetPos = iter.next ();
        nextPosAfterRule = processRule(nextGetPos);
    }
}
case:END LOOP
{
    endFlag.set(true);
    nextPosAfterRule = iter.next ();
}
}
case:UPDATE
{
    doUpdate();

    save position of current rule
    Rule nextRule = list.next();

    enter to a recursion with next rule from the list
    nextPosAfterRule = processRule(nextRule);
}
case:COMPARISON
{
    boolean isComp = doComparison();

    ast an entry accordingly to rule name in the Map ruleResultsData
    ruleResultsData.replace(rule.name, new CommonBo(isComp, Constants.AR_DATA_TYPE_ENUM));

    save position of current rule
    Rule nextRule = list.next();

    enter to a recursion with next rule from the list
    nextPosAfterRule = processRule(nextRule);
}
case:IGNORE
{
    Don't do anything
    continue;
}
}
}
return nextPosAfterRule;
}
}

List<CommonBO> doGet(Rule rule, Map<Rule name, CommnoBO> ruleResultsData)
{
    create a complete pure SQL phrase from parts of rule
    String sqlStatement = buildCompleteSqlStatement(rule, ruleResultsData);

    convert a SQL phrase to an AR Qualifier
    Qualifier qualifier = convertSqlToQualifier(sqlStatement);

    fire a Qualifier and get results by a Qualifier from AR(CMDB)
    List<Entry> entries = runRule(qualifier);

    return convertEntryToCommonBO(entries);
}

void doUpdate(Rule rule, Map<Rule name, CommnoBO > ruleResultsData)
{
    create a complete pure SQL phrase from parts of rule
    String sqlStatement = buildCompleteSqlStatement(rule, ruleResultsData);

    convert a SQL statement to an AR Qualifier
    Qualifier qualifier = convertSqlToQualifier(sqlStatement);
}

```

```

        runRule(qualifier);
    }

    boolean doComparison(Rule rule, Map<Rule name, CommnoBO > ruleResultsData)
    {
        create string statement with replacing placeholders on values
        String expression = buildStatement(ruleResultsData, rule.expression);

        return runExpression(expression);
    }

    String buildCompleteSqlStatement(Rule rule, Map<Rule name, CommnoBO > ruleResultsData)
    {
        work by a rule type
        switch(rule.type)
        {
            case:GET
            {
                create string statement with replacing placeholders on values
                String whereStatement = buildStatement(ruleResultsData, rule.where);

            }
            case:UPDATE
            {
                create string statement with replacing placeholders on values; from WHERE rule's member
                String whereStatement = buildStatement(ruleResultsData, rule.where);
                create string statement with replacing placeholders on values from SET rule's member
                String setStatement = buildStatement(ruleResultsData, rule.set);

            }
            default:
            {
                return;
            }
        }

        linkage a complete pure SQL phrase from SELECT, FROM, WHERE rule's members
        add ignore case if the FROM value is exist in ignore rule
        return a complete pure SQL phrase
        return fullSqlStatement;
    }

    String buildStatement(Map<Rule name, CommnoBO> ruleResultsData, String statement)
    {
        for all placeholders what will be found in the statement
        do{
            get first replacement expression
            String placeHolder = findReplacementExpression(statement);

            If placeholder is not empty
            {
                get rule name from the replacement expression
                String ruleName = separateRuleName(placeHolder);

                get rule attribute name from the replacement expression
                String ruleAttr = separateRuleAttribute(placeHolder);

                get rule attribute value from the Map
                String attrValue = findRuleAttrValue(ruleResultsData , ruleName, ruleAttr);

                replace the placeholder with the rule attribute value
                statement = replaceExpression(statement, placeHolder , attrValue);

            }
        } while placeHolder is not empty

        return statement;
    }
}

```

SWLM Rule Definition Interface

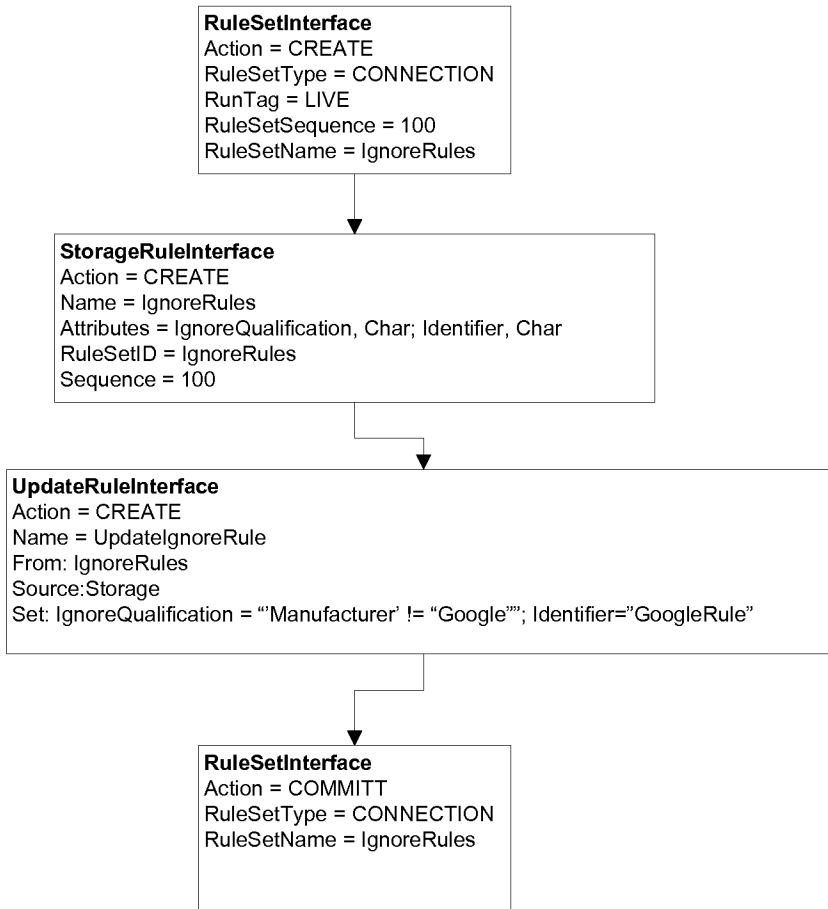
For the software license management administrative console we need to provide the ability to configure rules in a more user friendly way that does not require an end user to interact with the low level rules.

The concept is that we provide an interface that ask them some questions about connections and compliance and then make use of those answers to talk to the backend SWLM rules interfaces to generate the rules that are needed.

Lets look at the way we structure our rules to see how to best implement this functionality.

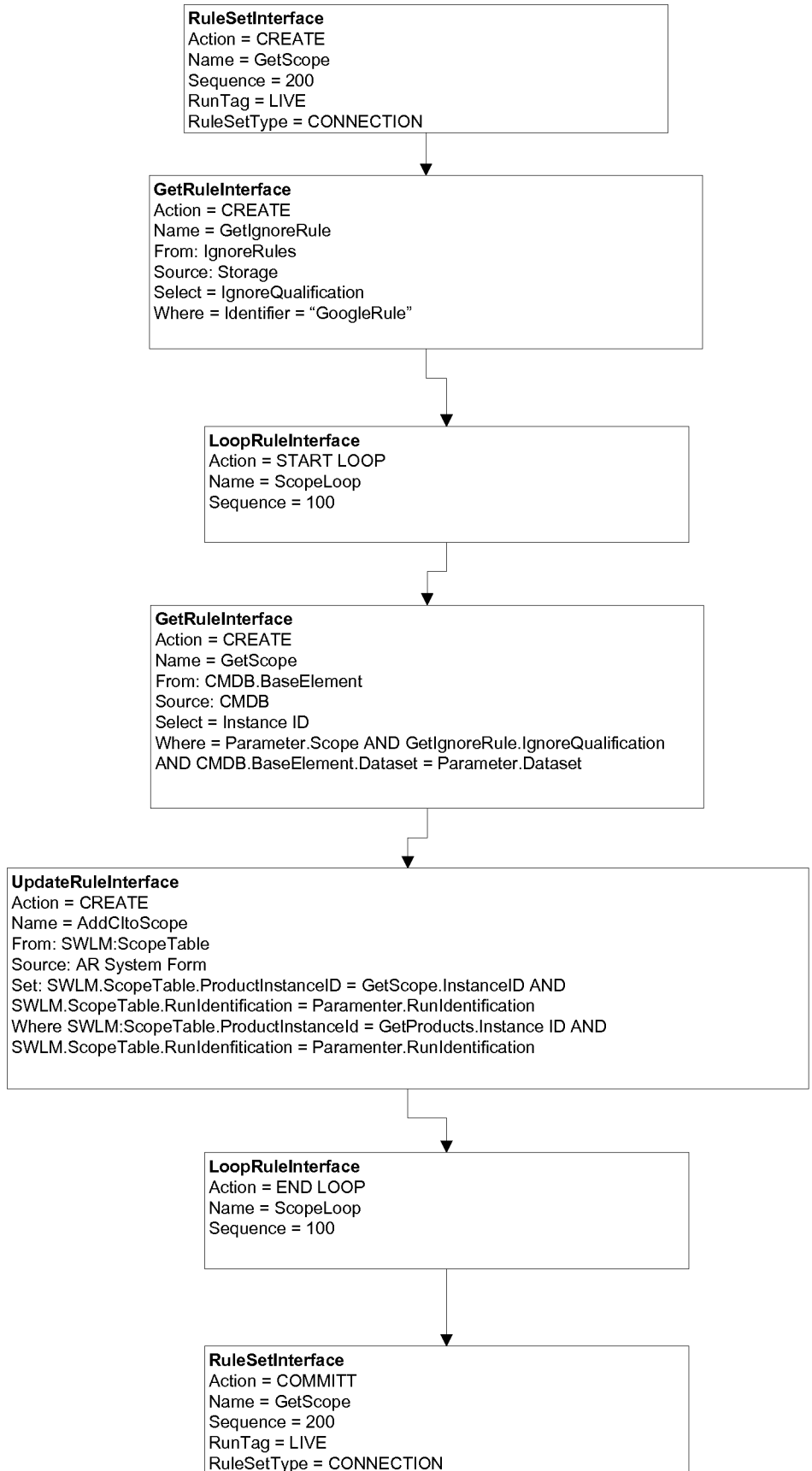
1. There is going to be common sets of rules for dealing with ignore and scope. These rule sets should be provided out of the box. For the ignore rule set we should provide an ninteface in front of it to continue to add to the rule set based on input from the user.

Below is an example of the calls to the interface for that would be needed to implement an ignore rule.



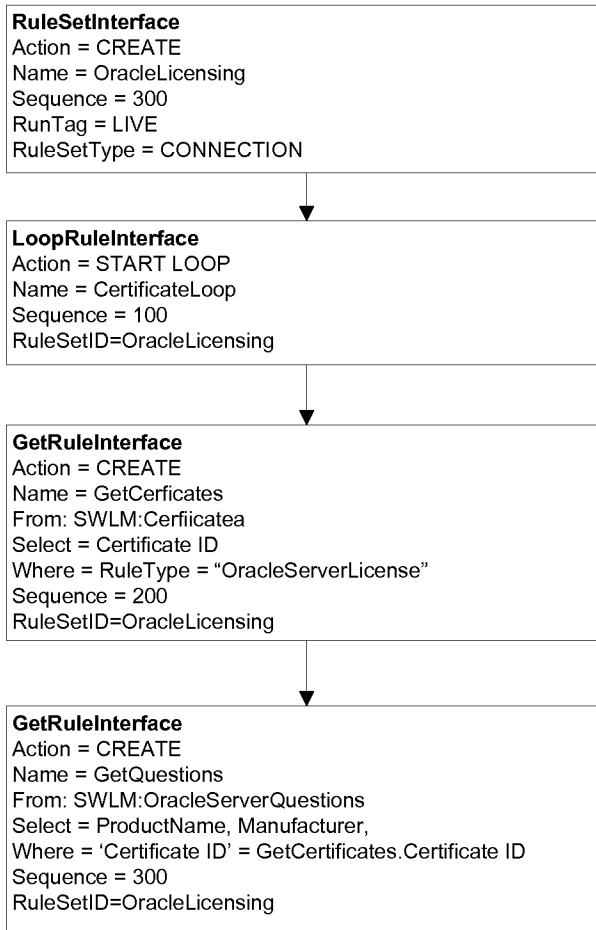
There is also a rule set for getting scope. What this ruleset does is take a parameter that is passed in at runtime, and the results of the ignore qualifications and make use of them to query the base element class of the CMDB to get a list of CIs that are needed to be processed.

If the engine is run as a result of a reconciliation rule, the scope would be related to the reconciliation run so that only CIs that have been updated as a result of the reconciliation run will be evaluated. The scope ruleset is a ruleset we should just provide out of the box and it should be used by all other rulesets.



2. The next set of rules that we need to worry about are the rulesets which are going to be specific to the rule types. For example, for PerInstance licenseing we would come up with a ruleset that interacts with the questions forms that are tied to certificates for PerInstance related license management.

When we look at the structure of these rule sets there are a number of rules which are going to be common for all license types. These are below:



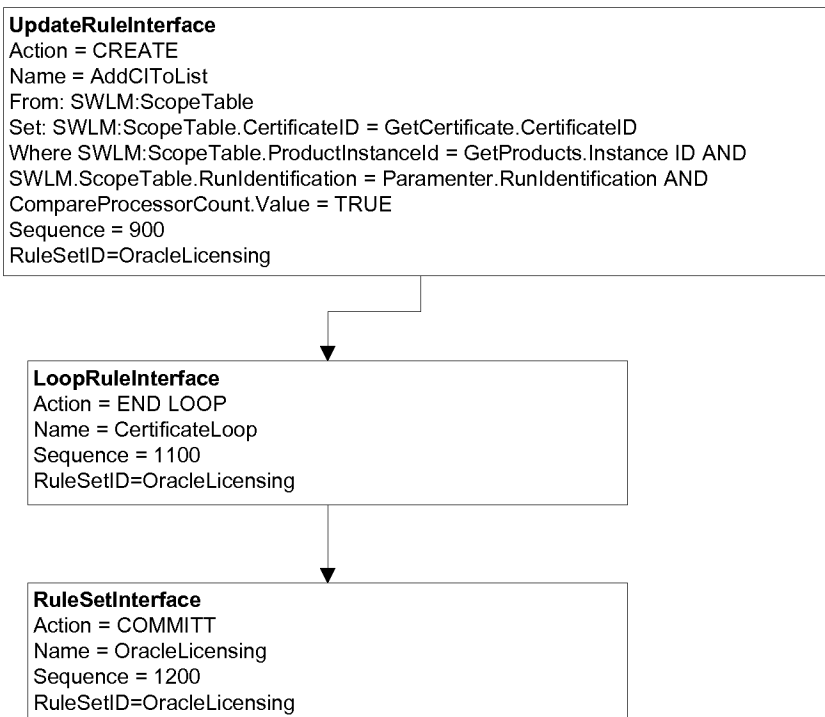
- The first rule here is the rule that starts the creation of the ruleset.
- The second rule starts a loop so we can loop through a set of certificates.
- The third rule gets the list of certificates for a particular rule type. The only difference in this rule will be the qualification. (RuleType = "OracleServerLicense")
- The fourth rule gets the data from the question form. The things that would be different in this rule would be the form name and the fields we are selecting. The qualification would be the same.

After you get through the first set of rules, the rest of the rules in the ruleset would be dealing with querying the CMDB based on the data that was pulled from the questions form. For example the following rule would be the fifth rule and it queries the product table to get the list of CIs from the CMDB that match the qualification built from the data in the questions form.

```

GetRuleInterface
Action = CREATE
Name = GetProducts
From: CMDB.Product
Select =Instance ID
Where CMDB.Product.Manufacturer = GetQuestions.Manufacturer AND
      CMDB.Product.ProuctName = GetQuestions.ProductName AND
      CMDB.Product.Dataset = Parameter.Dataset AND Parameter.Scope
AND IgnoreRules.IgnoreQualification
Sequence = 500
RuleSetID=OracleLicensing
  
```

After the querying of the data, the next set of rules would be used to create the scope list. This could either be done via pushing data to an AR Form, or pushing data to a memory structure. (The final implementation details are being worked out 3/12/08) These set of rules are also going to be the same for every ruleset.



Proposed Solution

To help a user manage the creation of the rulesets and manage the rulesets that we ship out of the box, the design is to have a UI which provides a user friendly interface to the generation of the rules. If you look at the description of the rules above you will see that a large set of these

rules are going to be common, with the differences being things that can be captured by asking a user questions, rather than having them go to a low level rule interface.

For connection rules, the interface would look something like this:

<<< NEED TO PROVIDE MOCKUP >>>>

The user would start from a form that would help them manage ruletypes, since this is the context in which the user is going to be building or managing the rules.

A user would either pick an existing ruletype, or create a new ruletype. If I select to create a new rule type, the interface will ask me a few questions about the ruletype.

1. The ruletype name.
2. The name of the connection form.
3. The name of the runtime data form.

When the user presses the next button there will be a filter API called that will take a template form for questions, and a template form for runtime data, and copy them to new form names, and make sure all the common workflow is attached to these new forms. **(NOTE: There is another extention to this that I am defining which would provide the ability to also pick a simple model for questions which would let them use a generic form that already has 10 fields on it, and make use of data for the labels. We would then have rules that would go off of this generic form. This will be useful for those cases where users do not need a rich UI for the questions, data validation on the question, workflow on the questions etc.)**

The user would then need to make use of the AR Administrator tool to go to the form and add any fields that are specific to the ruletype for question and compliance related data.

The user would then come back to the creation dialog to indicate they are done creating the forms, and they would then go into a dialog that would help them build the rules.

This interface would be broken into two components. One would be focused on connection rules and one that is focused on compliance rules.

On the connection side there is some information that we already know. We know the rule type, because we are only going to be creating rules in a context of a ruletype. We also know the forms that hold the questions and runtime data because that is registered with the rule type.

So, what we would need to ask the user is:

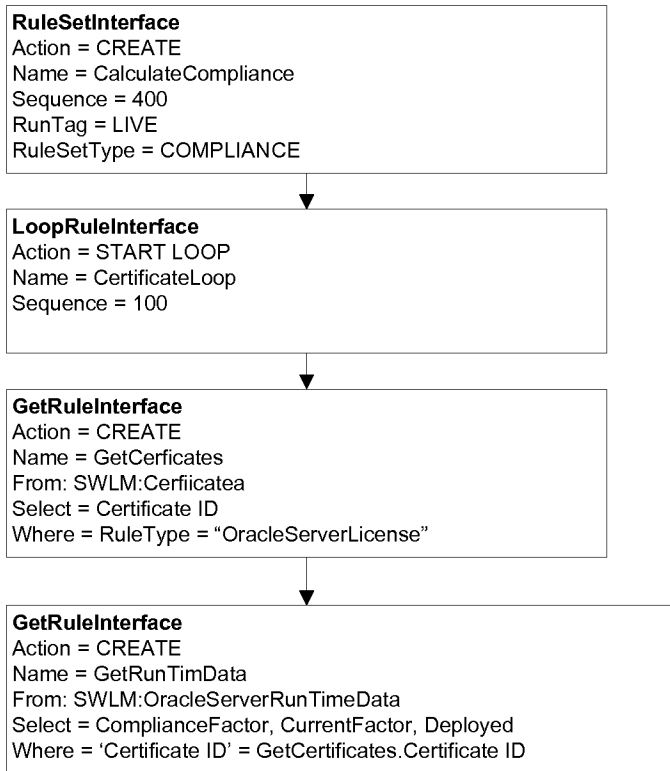
1. What fields they need to pull from the questions form to make use in the rules that will query the CMDB.
2. What is the query or set of queries that are needed to get the list of CIs.

<<< NEED TO SHOW A PROTOTYPE OF THIS>>>

The other set of rules that we need to have a simplified UI for is the compliance rules. The compliance rules do the work of calculating if a certificate is in compliance based on information about the license type. For example, compliance for PerInstance licensing would be based on #Deployed vs. #Purchased. Other licnese models will have different models that they use for managing compliance.

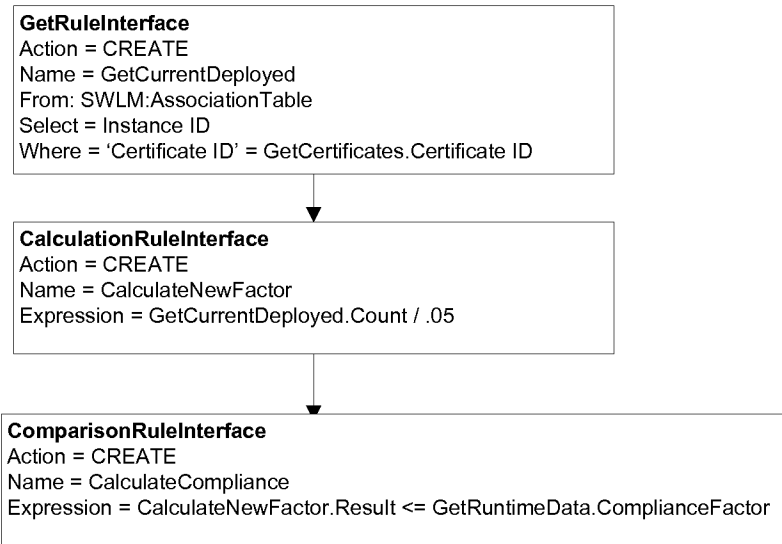
Like connection rules, compliance rules sets will have a set of rules which will be common to all compliance rules, and a set of rules which will require the end user to provide information so they can be built.

The common rules would be as follows. These rules setup the ruleset, start a loop of certificates and gets runtime data. We would need to ask the user information about what runtime data they want to pull since that will be unique on a per rule table basis, but we will be able to use that to build this first set of rules.



The rest of the rules in the ruleset will be specific to a type. For example, PerInstance licensing will need rules to look up current deployed from the association table, and compare with purchased.

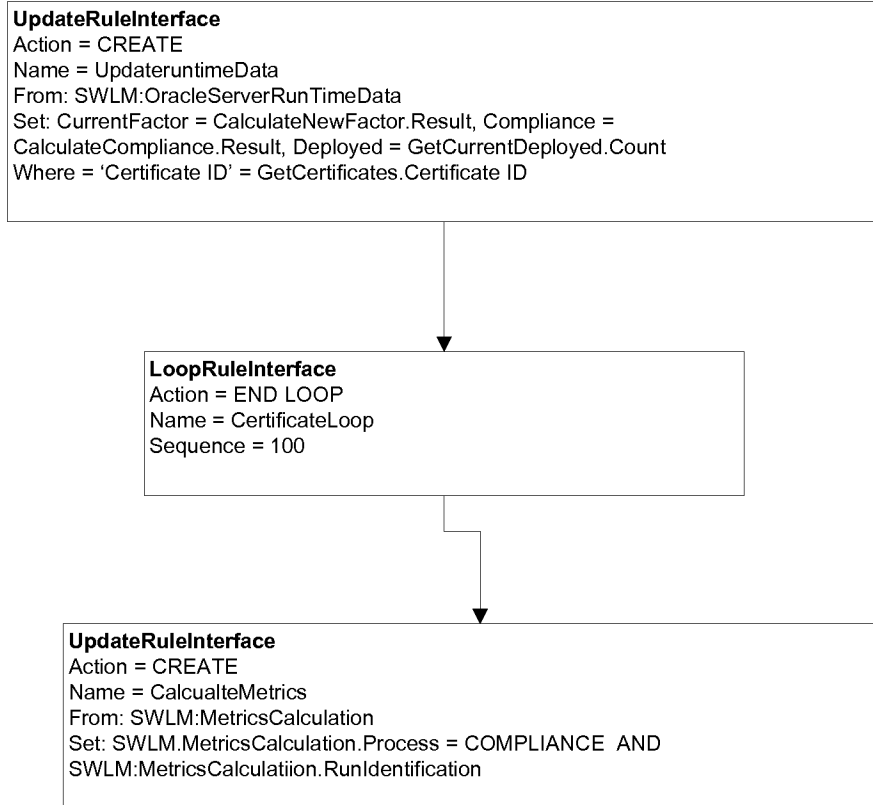
Oracle licensing may be more querying and have different types of calculations that need to be done. Here is a sample set of rules that do some license type specific calculations.



We need to look at how we can provide a user friendly interface to help build these rules.

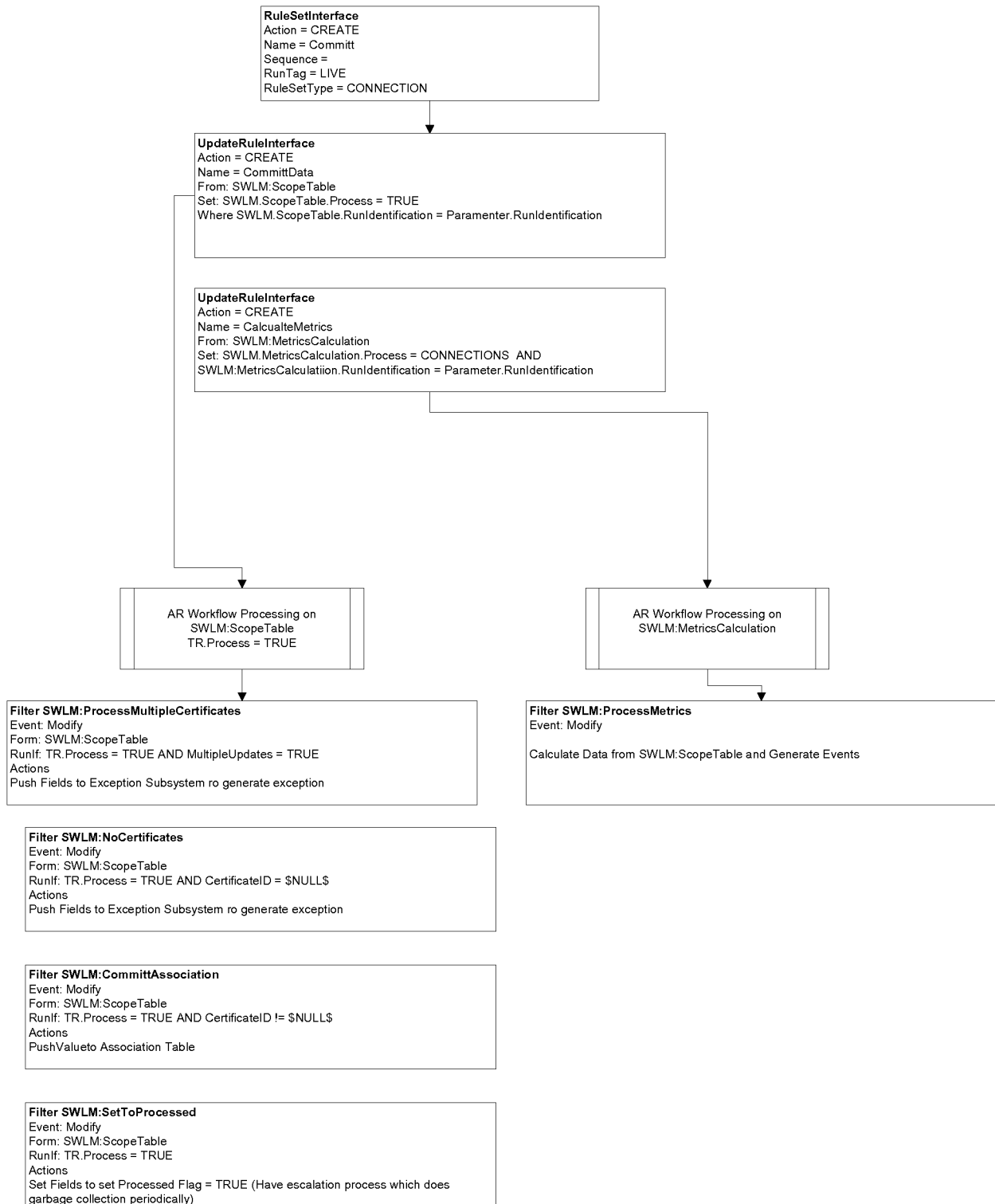
The last set of rules in the ruleset will be common also. These rules update the runtime data, and close the transaction. We would need to ask the user what the fields to update runtime data are, and from there we can build the rules.

Example closing rules



Other Common Rule Sets

After the connection rules there is another set of common rules that will be run for committing the data to the database. These should be provided out of the box. The final implementation will be dependant on if there is memory storage functionality available.



Rule Definition UI

BMC REMEDY IT SERVICE MANAGEMENT - Software License Management Welcome Help

Manage License Types

Create New License Types

Step 1 - Select a License Type

License Type:

Step 2 - Define Provide Information Model

Model:

Description:

Frequency:

Step 3 - Generate Rules

BMC REMEDY IT SERVICE MANAGEMENT - Software License Management Welcome Help

Rule Generation Wizard

Create Connection Rules

Step 1 - What Fields to Use?

Fields to Use

Field Name:

Selected Fields

Field Name:

Step 2 - Define Queries to Select CIs

Data Source:

Sequence:

Form Name:

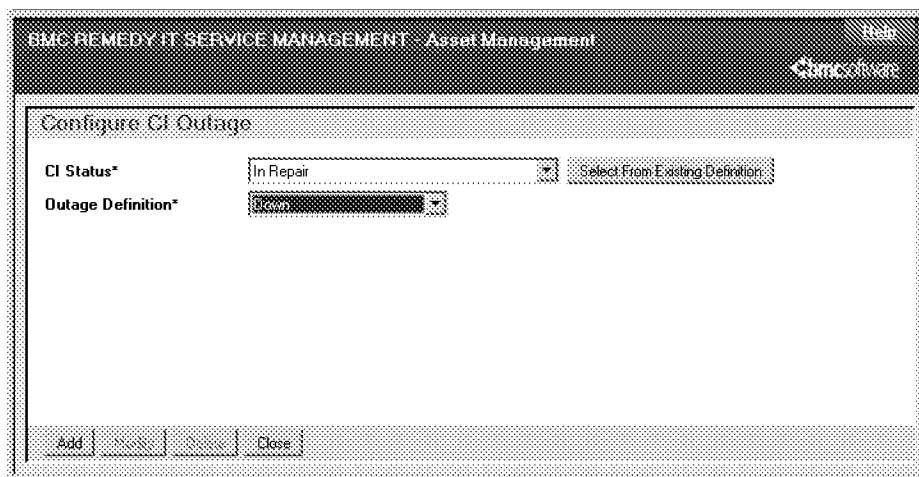
Qualification:

Fields to Select

Sequence:

The Configure CI Outage form appears as shown in Figure 15-8.

Figure 15-8: Configure CI Outage form

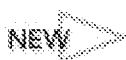


- 3 Form, from the CI Status list, select the CI status for which you want to configure unavailability.
- 4 From the Outage Definition list, select Down or Up.
- 5 Click Add.

TIP

You can click Select From Existing Definition to select a previously defined unavailability status from the list displayed.

Configure License Type



License types are used for software license management. The license type wizard allows administrators with Contract Config or Asset Config permissions to create custom license types for license certificates. After the license type wizard has been completed and the license type has been saved, the license types are used when a contract manager or software asset manager creates certificates.

When the license engine runs, company and product information and answers to questions supplied when the certificate was created are used to determine which CIs should be connected to the license certificates. Compliance is also calculated based on the answers to questions supplied and the configured compliance actions which act on these values.

For more information about configuring the license engine to run jobs, see “Configure License Engine Run” on page 458 and “License Engine Configuration” on page 455. For more information about running jobs, see the *BMC Remedy Asset Management User’s Guide*.

You can use license types to specify the following information:

- What questions a contract manager or software asset manager must answer when adding a new software license certificate.

- How connection questions are used to select CIs from BMC Atrium CMDB. For more information about the BMC Atrium CMDB, see the *BMC Atrium CMDB Administrator's Guide*, the *BMC Atrium CMDB User's Guide*, and the *BMC Atrium CMDB CDM Common Data Model Diagram*.
- How compliance is computed based on compliance questions and actions specified during license type creation, such as relating CIs to a license certificate or creating exceptions.

You cannot modify license types after they have been used to create certificates.

..... **TIP**

For examples of how to configure your own license types, see Appendix B, "Create license type examples".

.....

Default system license types

When the BMC Remedy Asset Management application is installed, the following default license types are already configured:

- Enterprise (company-based)
- Site
- Per instance
- Per copy per device
- Per copy
- BMC Remedy AR System fixed and floating

..... **NOTE**

All license types have to be enabled, including the license types shipped with the BMC Remedy Asset Management application. For more information, see "Enabling and disabling license types" on page 448.

.....

If you want to use additional license types, follow the instructions in "Creating custom license types" on page 426 to create them.

Default behavior for customized license types

For any default or custom license type, the system uses default behavior when generating the rules for various license types. This default behavior is used whether or not you set up definitions in the license type wizard.

This default behavior for *connection* criteria uses a certificate's company and product categorization criteria to query certificates matching that license type and determine which CIs can be connected to a certificate. The default behavior is used in addition to any questions that are specified in the license type wizard for that license type.

This default behavior for *compliance* criteria queries certificates that match a license type and tests the result of the query against compliance criteria.

If you need more functionality in addition to the default behavior, you have to create a custom license type. You can configure additional connection behavior, or compliance actions, or both. The custom license type can be configured in either basic or advanced modes as described in “Basic and advanced mode.” If you want to use additional license types, follow the instructions in “Creating custom license types” on page 426 to create additional license types.

Table 15-1 explains how the default license types work.

Table 15-1: System license types

Contract type	Description
Enterprise	<p>An enterprise license type licenses all of the software from a manufacturer that is being used by a given company. There is one enterprise license per company. Anyone in the company can use the license. This license type is set to non-groupable.</p> <p>Connection</p> <p>Uses the default certificate criteria for connections. See “Default behavior for customized license types” on page 418.</p> <p>Compliance</p> <p>One computed compliance question is specified for number of licenses deployed. The certificate for this license type is always compliant.</p>
Site	<p>A site license type licenses all of the software from a manufacturer that is being used by a given site. A site license applies to a single site within a company. If a site license applies to multiple sites within a company, the contract manager or software asset manager can add a site license certificate for each site. This license type is set to non-groupable.</p>

Table 15-1: System license types (Continued)

Contract type	Description
Site (continued)	<p>Connection</p> <p>Uses the default certificate criteria for connections. See “Default behavior for customized license types” on page 418.</p> <p>In addition, one connection question exists for this license type, which is “Enter Certificate Site?”. When the license type is created, this question maps to the site of the computer system on which the product is installed.</p> <p>For each product that matches the default certificate criteria for connections, the site of the computer system on which the product is installed is used to connect the product to the license certificate to that site.</p> <p>For example, a contract manager creates a license certificate with the default certificate criteria for connections:</p> <ul style="list-style-type: none"> ※ Company = Calbro Services ※ Manufacturer = Microsoft ※ Product = Visio <p>The contract manager specifies this Site license certificate is for Frontoffice Support.</p> <p>A copy of Microsoft Visio is installed on a computer in Frontoffice Support. That product is not connected to the license certificate.</p> <p>A computer in Backoffice Support Services has both Microsoft Visio and Microsoft Word installed. Microsoft Visio is connected to the site license, but Microsoft Word is not (perhaps it is connected to an Enterprise license).</p> <p>Compliance</p> <p>One computed compliance question is specified for number of licenses deployed. The certificate for this license type is always compliant.</p>
Per instance	<p>Each instance of the license is counted as one license consumed. Any instance of software found requires a license. This license type is set to groupable.</p> <p>Connection</p> <p>Uses the default certificate criteria for connections. See “Default behavior for customized license types” on page 418.</p> <p>Compliance</p> <p>The following compliance question is asked:</p> <ul style="list-style-type: none"> ※ Number of licenses purchased? ※ Breach warning level 1? ※ Breach warning level 2? ※ Number of licenses deployed (computed question)

Table 15-1: System license types (Continued)

Contract type	Description
Per instance (continued)	<p>The answers to this question are used to:</p> <ul style="list-style-type: none"> ⌘ Determine the number of software instances related to the certificate ⌘ Set the certificate to not compliant, if the number of software instances related to a certificate is greater than the number purchased. <p>Example</p> <p>Two instances of BMC Atrium CMDB Enterprise Manager were found. In this situation, two licenses are required.</p>
Per copy per device	<p>The number of copies per device is counted as one license consumed. If the number of copies per device is exceeded, the license is out of compliance. The per device number specified for a license type determines how many copies can be installed on each device. This license type is set to groupable.</p> <p>Connection</p> <p>Uses the default certificate criteria for connections. See “Default behavior for customized license types” on page 418.</p> <p>Compliance</p> <p>The following compliance questions are asked:</p> <ul style="list-style-type: none"> ⌘ Number of licenses purchased? ⌘ Breach warning level 1? ⌘ Breach warning level 2? ⌘ Number of copies allowed per device? ⌘ Number of licenses deployed (computed question). <p>The answers to these questions are used to compute compliance:</p> <ul style="list-style-type: none"> ⌘ If any devices have more than the allowable number per device, the certificate is marked as not compliant. ⌘ Also, if the number of software licenses related to a certificate is greater than the number purchased, the certificate is marked as not compliant. <p>Example</p> <p>Two copies of Microsoft Word were found on a computer, but only one license exists. If the number of copies allowed on a device is one, this is not compliant.</p>
Per copy	<p>This license type is per copy per user. Each unique user (since, for example, there could be two software items linked to one user) is counted as one license consumed. If a user is not linked to a license, it is assumed that one license is consumed. This license type is set to groupable.</p> <p>Connection</p> <p>Uses the default certificate criteria for connections. See “Default behavior for customized license types” on page 418.</p>

Table 15-1: System license types (Continued)

Contract type	Description
Per copy (continued)	<p>Compliance</p> <p>Compliance is computed based on the number of unique individuals using a software product. The following compliance question will be asked:</p> <ul style="list-style-type: none"> ※ Number of licenses purchased? ※ Breach warning level 1? ※ Breach warning level 2? ※ Number of licenses deployed (computed question). <p>The answer to these questions are used to determine the number of unique individuals connected to a product. If the number counted is greater than the number of licenses purchased, the license certificate is set to not compliant.</p> <p>Example</p> <p>Two copies of an application were found on separate computers owned by one user and only one license exists. This situation is compliant, because only one license is required.</p>
BMC Remedy AR System fixed and floating	<p>The number of fixed and the number of floating licenses are compared against the number that is entitled. This is a demonstration of the advanced model. To walk through the creation of this license type as an example of the advanced model, see “BMC Remedy AR System fixed and floating license type example—advanced mode” on page 520.</p>

What is grouping?

If you have multiple certificates in different contracts, you can group these certificates under a master certificate. Only certificates belonging to the same product categorization should be grouped. For example, you might have a site license for Microsoft Visio for one location and 50 per instance licenses. Or, you might have two different site licenses. You can use the default license types for grouped certificates, or you can create a custom license type. For more information, see the *BMC Remedy Asset Management User's Guide*.

Basic and advanced mode

You use the following modes when creating custom license types:

- **Basic**—allows you to create connection questions and map these based on the answers entered for the questions when a license certificate is created. In addition to configuring connection information, you can also create compliance questions and actions to determine if you license certificates and CIs are compliant with your licensing agreements, for example, to determine whether or not you have deployed more licenses than you have purchased.

For more information about creating a license type in basic mode, see “Basic mode” on page 427.

-
- **Advanced**—uses template forms and allows you to add your own fields. Then the fields are mapped using the license type wizard. This mode allows more flexibility when creating a custom license type.

For more information about creating a license type in advanced mode, see “Advanced mode” on page 444.

What are connection questions?

Connection questions (in basic mode) and customized connection forms (advanced mode) and mappings define the criteria by which CIs are connected to certificates. A default set of connection behavior is used regardless of whether connection questions or forms are configured for a license type. See “Default behavior for customized license types” on page 418. If the default behavior is sufficient, connection questions, or forms, do not have to be configured for a custom license type.

To determine which CIs should be connected to a license certificate, the license engine uses the answers supplied to connection questions or supplied in connection forms at the time a license certificate is created. For more information about creating license certificates, see the *BMC Remedy Asset Management User's Guide*.

What are compliance questions and actions?

Compliance questions and actions work on the retrieved connection information to determine if licenses are compliant with their license contract. You use the license type wizard to configure compliance questions (for basic mode) or a compliance form (for advanced mode).

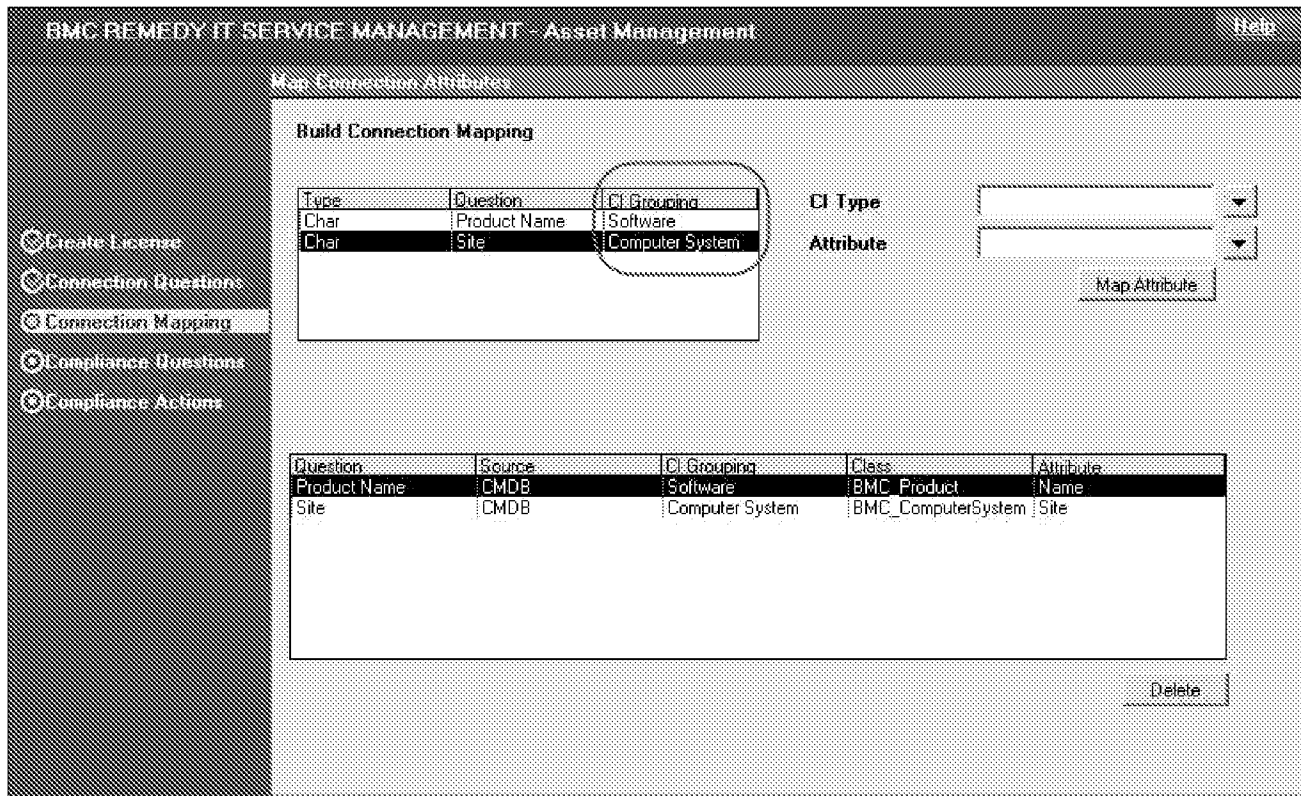
You also configure actions that test the compliance of CIs connected to licenses based on values entered at the time the license certificate is created. For example, you can determine if there are more deployed licenses than were purchased for a particular site.

What is a complex query?

CIs are grouped in a hierarchy where one main class contains sub-classes, which are referred to as the CI grouping. The CI Grouping field on the Connection Questions page of the license type wizard contains both the main class (which populates the CI Type field on the Map Connection Questions page of the license type wizard) as well as the sub-classes (which populate the Attribute field on the Map Connection Questions page of the license type wizard). You are creating a complex query if you build connection questions from a main class and a sub-class or two separate sub-classes. For more information about creating a complex query in the license type wizard, see step 4 and step 5 on page 430. For more information about the CMDB models on which license type are based, see *Using the CDM and Extensions to Model Business Entities White Paper*.

Figure 15-9 shows a complex query being configured in the Map Connection Attributes page of the license type wizard.

Figure 15-9: Map Connection Attributes page showing complex query being created



Running the license engine

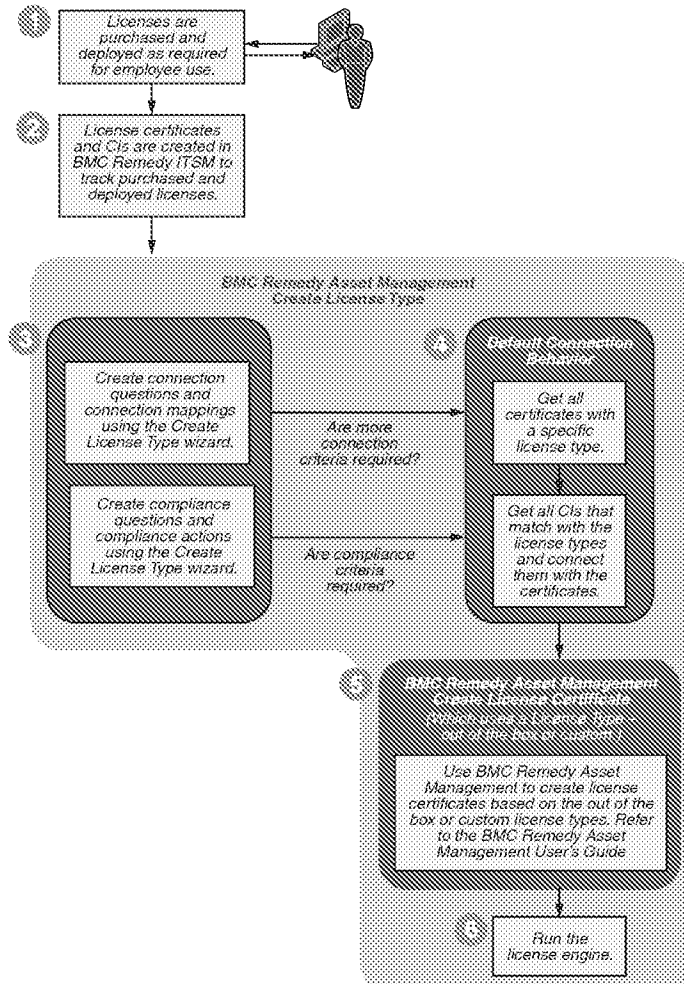
After license certificates are set up with the appropriate license types, you can run the license engine to connect CIs and compute compliance. You can schedule the license engine to run immediately, at a scheduled time, or after reconciliation. Licensing jobs are run from the Manage License Jobs Console. For more information about the Manage License Jobs Console, see the *BMC Remedy Asset Management User's Guide*.

Certain parameters that control how the license engine runs can also be configured. For more information about configuring the license engine, see "Configure License Engine Run" on page 458 and "License Engine Configuration" on page 455.

Overview of the create license type process

Figure 15-10 shows the process that you follow to create a license type.

Figure 15-10: Create license type process flow



Creating custom license types

NOTE

When you create a license type, it is created in draft mode. You need to enable it before you can use it. For more information about enabling license types, see “Enabling and disabling license types” on page 448.

► **To configure a new license type**

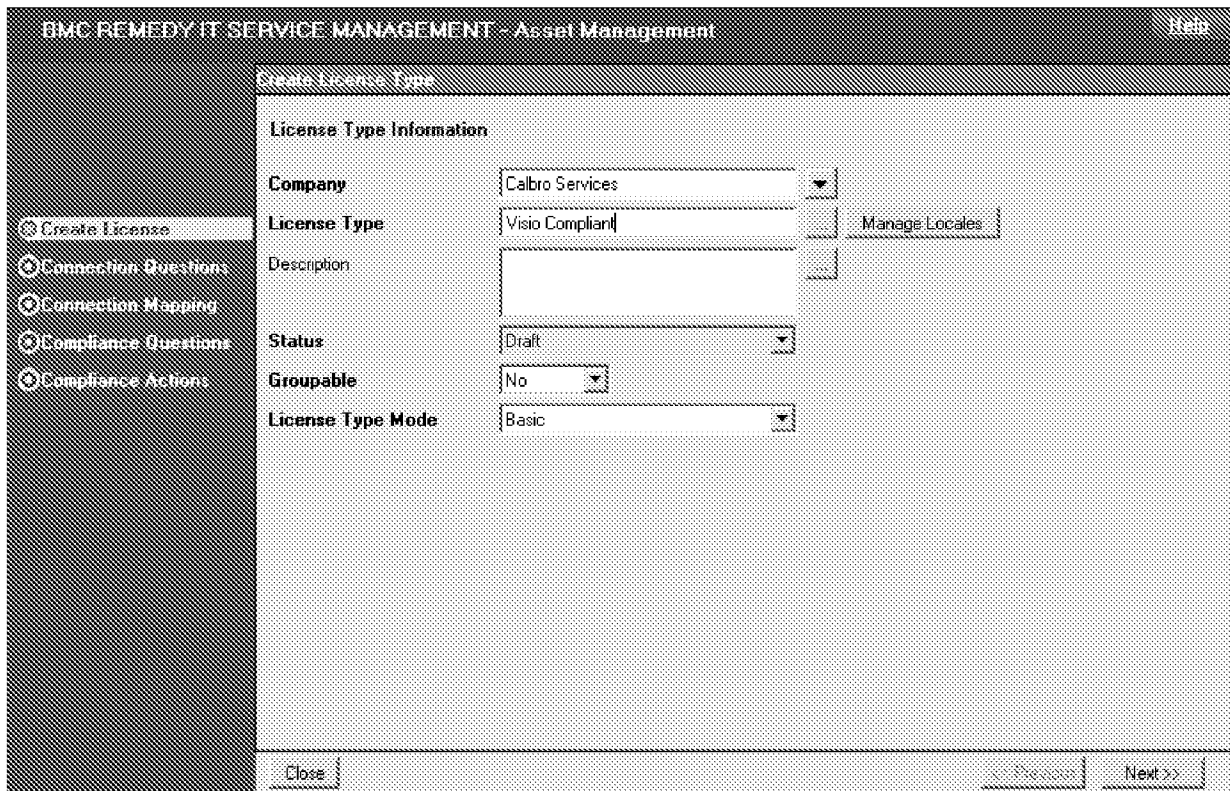
- 1 From the Application Administration Console, click the Custom Configuration tab.
- 2 From the Application Settings list, choose Asset Management > Advanced Options > Configure License Type, and then click Open.

The Configure Licenses form lists the current license types.

- 3 Click Create.

The Create License Type page of the license type wizard is displayed as shown in Figure 15-11.

Figure 15-11: License type wizard (page 1) - Create License Type



- 4 Specify the following fields, and then click Next.

TIP

You can click Manage Locales to enter license type and description in different languages.

Field	Description
Company	Select the company that can use this license type. If this license type can be used by all companies, select -Global-.
License Type	Enter the name of the license type.
Status	The status has a default value of Draft. This value cannot be changed. After the license type has been created, it needs to be activated as described in “Enabling and disabling license types” on page 448.
Description	Optional. A description of the license type being created.
Groupable	Specify whether certificates of this type are to be grouped under a master certificate based on license type and product categorization. Note: If you select Yes here, you later use the summable and distributable options for compliance questions. For more information about using the summable and distributable options available, see step c on page 432.
License Type Mode	Select Basic or Advanced. The basic mode uses a wizard to guide you through creating the license type. <ul style="list-style-type: none"> ※ Basic—Continue with the steps described in “Basic mode” on page 427. ※ Advanced—Continue with the steps described in “Advanced mode” on page 444.

Basic mode

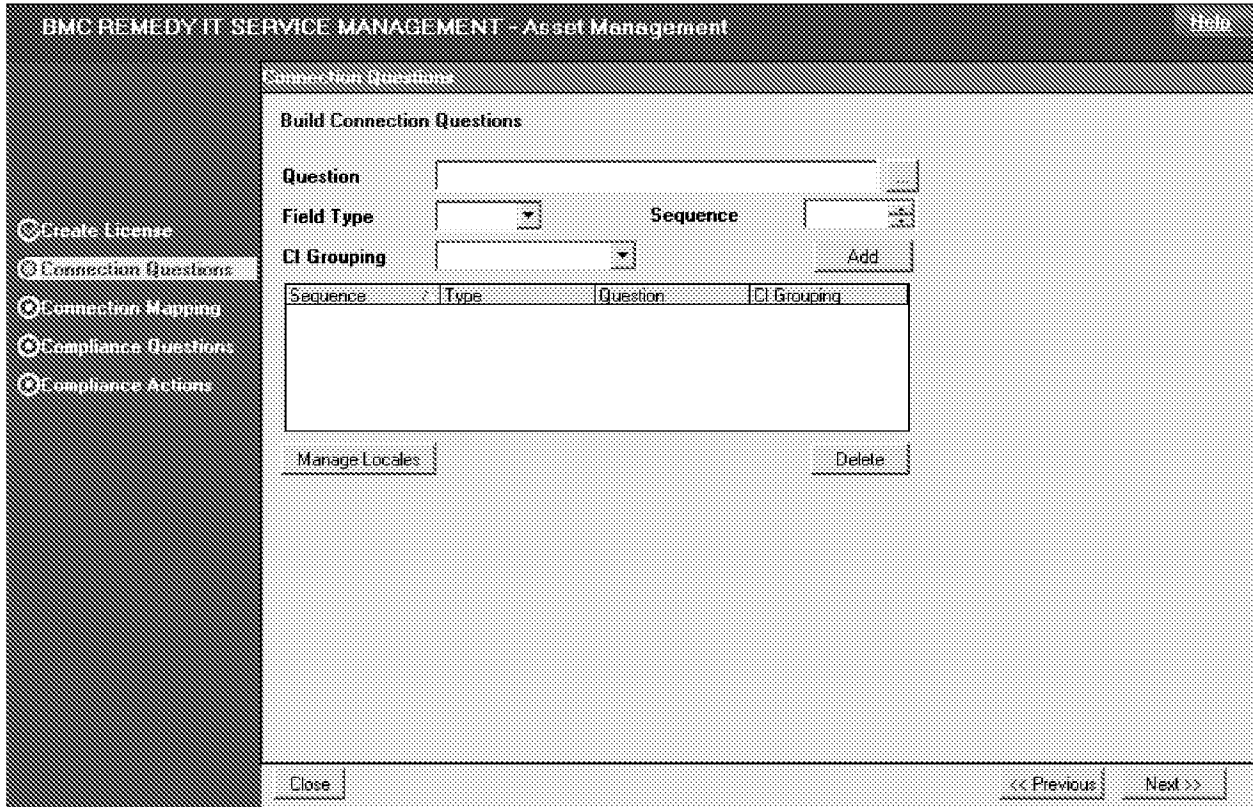
This section continues with the rest of the steps required to create a license type in basic mode. For an example of how to create a license type in basic mode, see “Site license type example—basic mode” on page 498 and “Per instance license type example—basic mode” on page 508.

► **To continue creating a license type using basic mode**

- 1 You should already have completed the steps in “Creating custom license types” on page 426.
- 2 In the License Type Mode, Basic should be selected. If it is not, select it now.
- 3 Click Next.

The answers to the connection questions are used to return a list of CIs that match certain criteria and the Connection Questions page of the license type wizard for basic mode is displayed as shown in Figure 15-12.

Figure 15-12: License type wizard (page 2) - Connection Questions



4 Specify the connection questions.

For each connection question, perform the following steps:

- a Type a question in the Question field.

The question name is the question that is displayed to the user who adds a license certificate. In the next stage of the license wizard, you map this question to an attribute in the BMC Atrium CMDB. The user’s answer to the question is used by the license engine in a query to BMC Atrium CMDB.

- b Select a field type of either Char or Int.

The field type determines whether the user enters a character or integer answer to the question.

- c Select the CI grouping:

- **Software**—The question is about the software being licensed.
- **Component**—The question is about a component on the computer system on which the software is installed.

-
- **Computer System**—The question is about the computer on which the software is installed.

Selecting the CI grouping for the question, also determines the CI type attribute you can select to map to it. For example, if the CI grouping question is set as Software, then you can only map attributes belonging to the software classes. The CI Type menu is dynamically populated when you select the question to be mapped.

- d Select the sequence in which the question is asked.
- e Click Add.

TIP

To delete a question, highlight it in the list and click Delete. If you want to modify a question, you must delete it and add a new question.

- f To enter questions in other languages, select a question and click Manage Locales.

The Manage Locales form displays the selected question. You can add translations of the question for other locales.

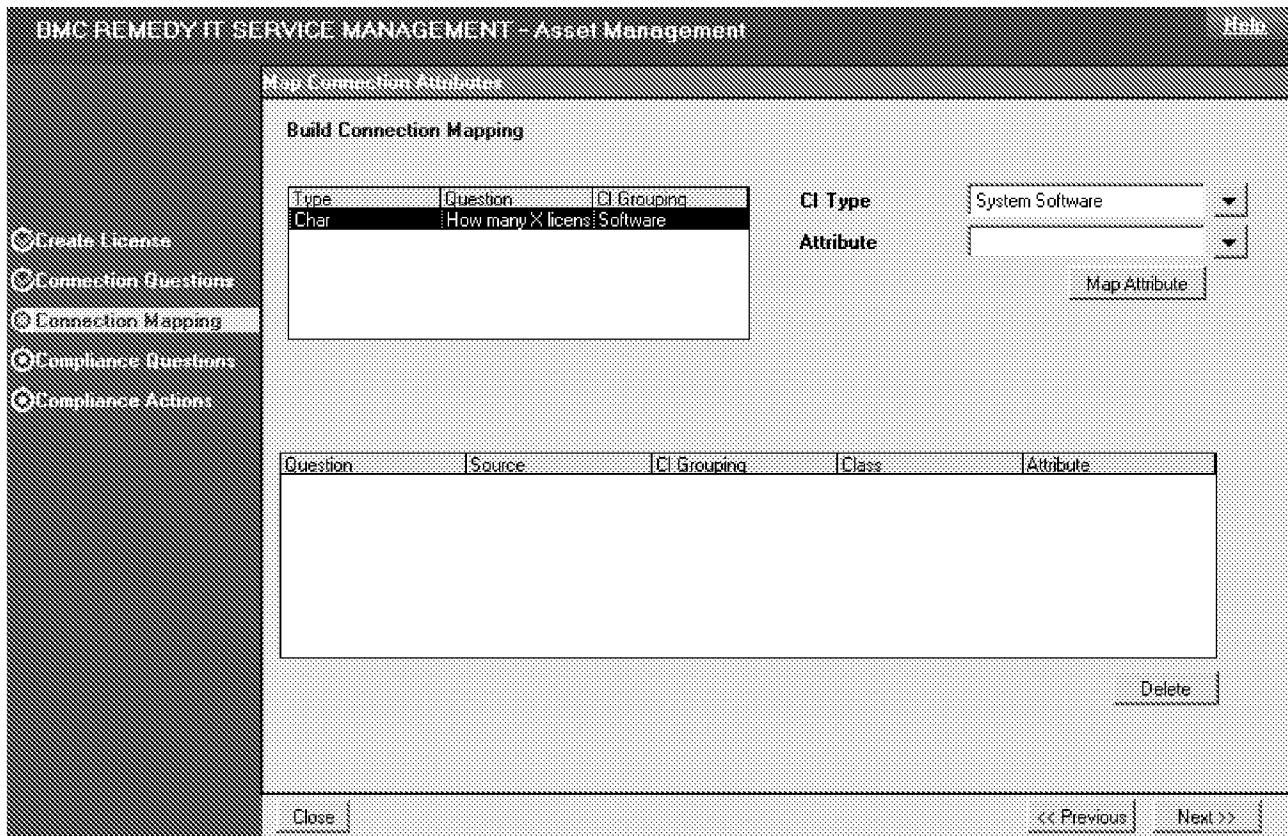
- g When you have finished adding questions, click Next.

The Map Connection Attributes page of the license type wizard for basic mode is displayed as shown in Figure 15-13.

TIP

The Map Connection Attributes page is displayed only if you define connection questions.

Figure 15-13: License type wizard (page 3) - Map Connection Attributes



- 5 Specify how to map the questions to CIs in BMC Atrium CMDB.

For each connection question, select the CI Type and Attribute and click Map Attribute. You must use the same CI Type for each connection question. The query to connect to the appropriate CI uses a logical AND between attributes.

NOTE

For any CI Type that you select, the Attribute field lists the DB names used in BMC Atrium CMDB forms.

TIP

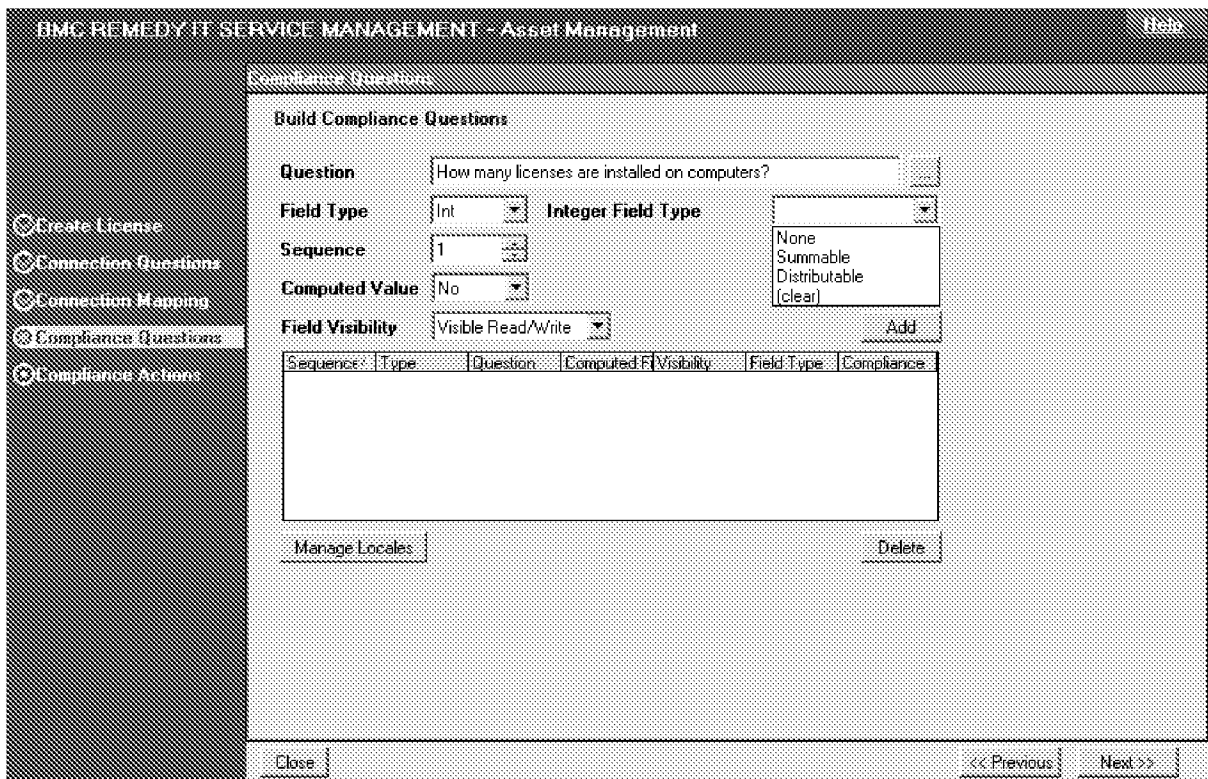
To delete a mapped attribute, select it from the list at the bottom of the Build Connection list and click Delete.

6 Click Next.

The compliance questions are used by the compliance actions to determine the compliance for the given license type. The CIs connected to a certificate and the number of CIs that are allowed to be connected are analyzed through these actions.

The Compliance Questions page of the license type wizard for basic mode is displayed as shown in Figure 15-14.

Figure 15-14: License type wizard (page 4) - Compliance Questions



7 Specify the compliance questions as follows:

- a Type a question in the Question field.

The question name is the question that is displayed to the person who adds a license certificate. In the next stage of the license type wizard, you build compliance actions to get, calculate, compare, or update values based on answers entered when a license certificate is created using this license type. The answers to the questions are used by the license engine in a query to BMC Atrium CMDB.

- b Select a field type of either Char or Int.

The field type determines whether the user enters a character or integer answer to the question.

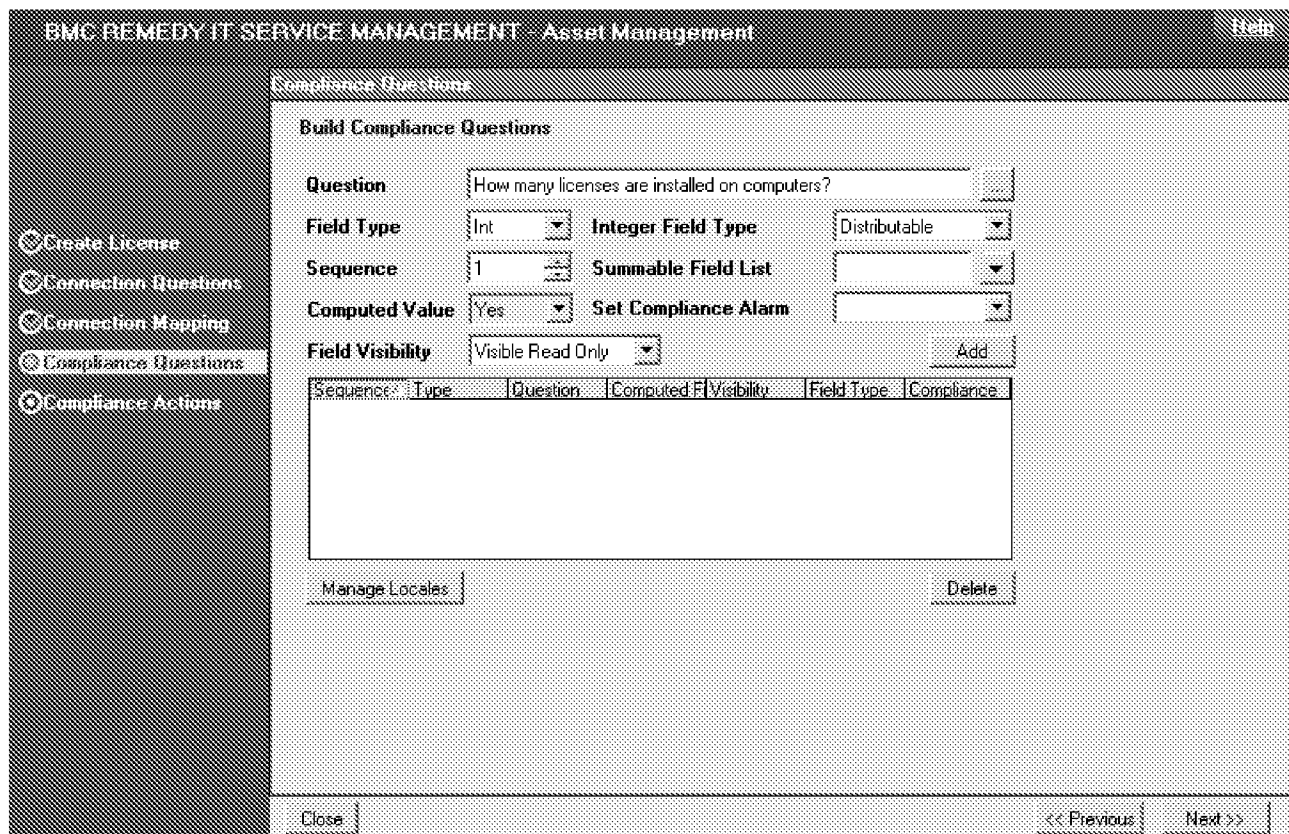
c If you specified Groupable on the Create License Type page of the license type wizard and Int as the field type here, select one of these values in the Integer Question Type field:

- ❖ **None**—indicates that the field is to accept an integer value that is operated on according to actions specified in the Build Compliance Actions page of the license type wizard.
- ❖ **Summable**—indicates that the field that you specify as summable gets rolled up from all the children to the master certificate.
- ❖ **Distributable**—Uses the summable amount and distributes the number to the children based on the sequence of the children and the value for each.

For example, on three separate occasions, 10 licenses, 5 licenses, and 20 licenses were purchased. When their certificates are grouped at the master level, 35 are displayed for summable. The distributable value that is displayed shows the value for each certificate. If a total of 45 CIs are found connected to the certificate, 10 are assigned to the first child certificate, 5 to the second, and the remaining 30 for the last. (Amounts are distributed back to the child certificates based on the sequence in which they were grouped under the master certificate.) Since the last certificate gets assigned 30 while it had only purchased 20, it is marked as not compliant. The master certificate is also marked as not compliant.

- d If you selected Distributable in step c above, these two additional fields appear as shown in Figure 15-15:
- **Summable Field List**—Select the summable field based on which the distributable field should divide the totals amongst the children certificates.
 - **Set Compliance Alarm**—Select whether or not you want to set a compliance alarm. If set to Yes, this distributable field is used to trigger compliance on the children certificates.

Figure 15-15: Summable Field List and Set Compliance Alarm fields



- e Specify whether the question uses a computed value.
- You can specify rules on the next page of the license type wizard to compute values.

- f Select the field visibility:
 - ⌘ **Visible Read / Write**—identifies fields that are updatable from the certificate.
 - ⌘ **Visible Read Only** —categorizes fields that do not need to be updated from the certificate, such as those retrieved for a compare or computation. For example, this attribute can be used for the results of CMDB queries.

Selecting this value results in a dimmed, uneditable field when you use this license type to create a license certificate from the Software Asset Management (SAM) Console.
 - ⌘ **Hidden**—Used to categorize fields that are used in an operation but do not need to be seen in the certificate.
- g Select the sequence in which the question is asked.
- h To enter questions in other languages, select a question and click Manage Locales.

The Manage Locales form displays the selected question. You can add translations of the question for other locales.
- i Click Add.

..... **TIP**

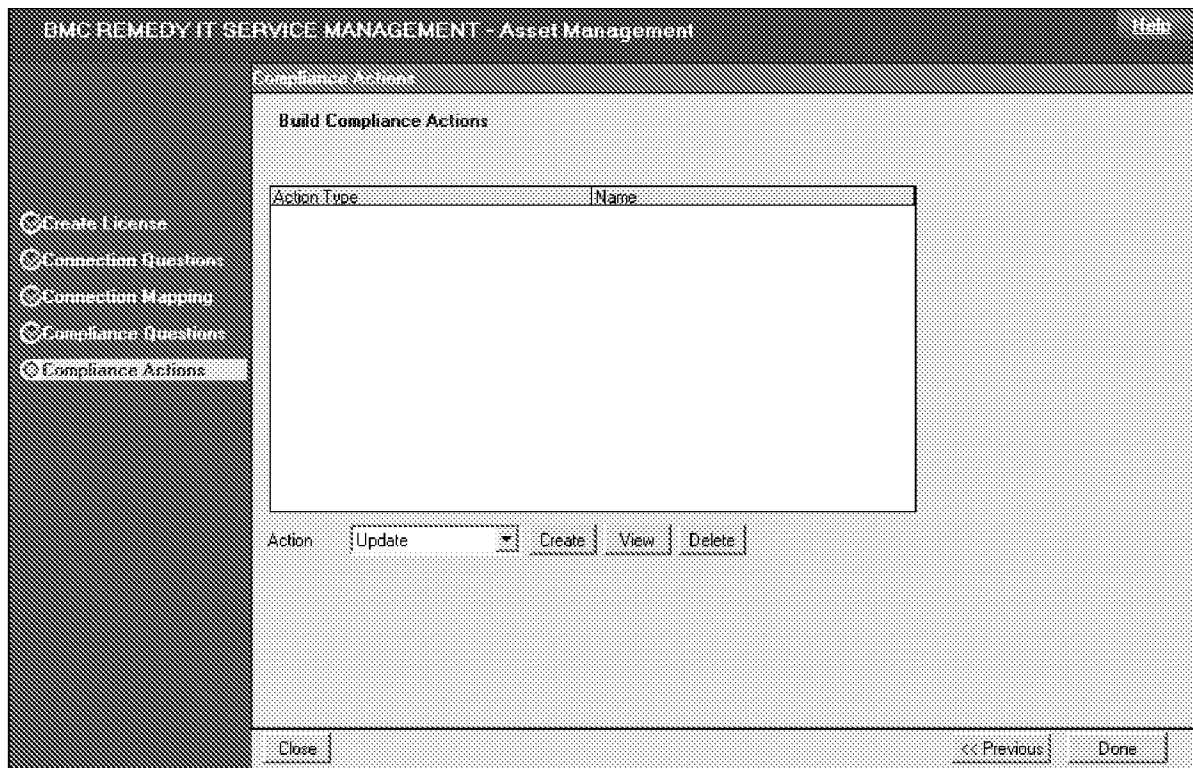
To delete compliance questions, select them from the list at the bottom and click Delete.

.....

- j When you have finished adding all of the compliance questions, click Next.

The Compliance Actions page of the license type wizard for basic mode is displayed as shown in Figure 15-16.

Figure 15-16: License type wizard (page 5) - Compliance Actions



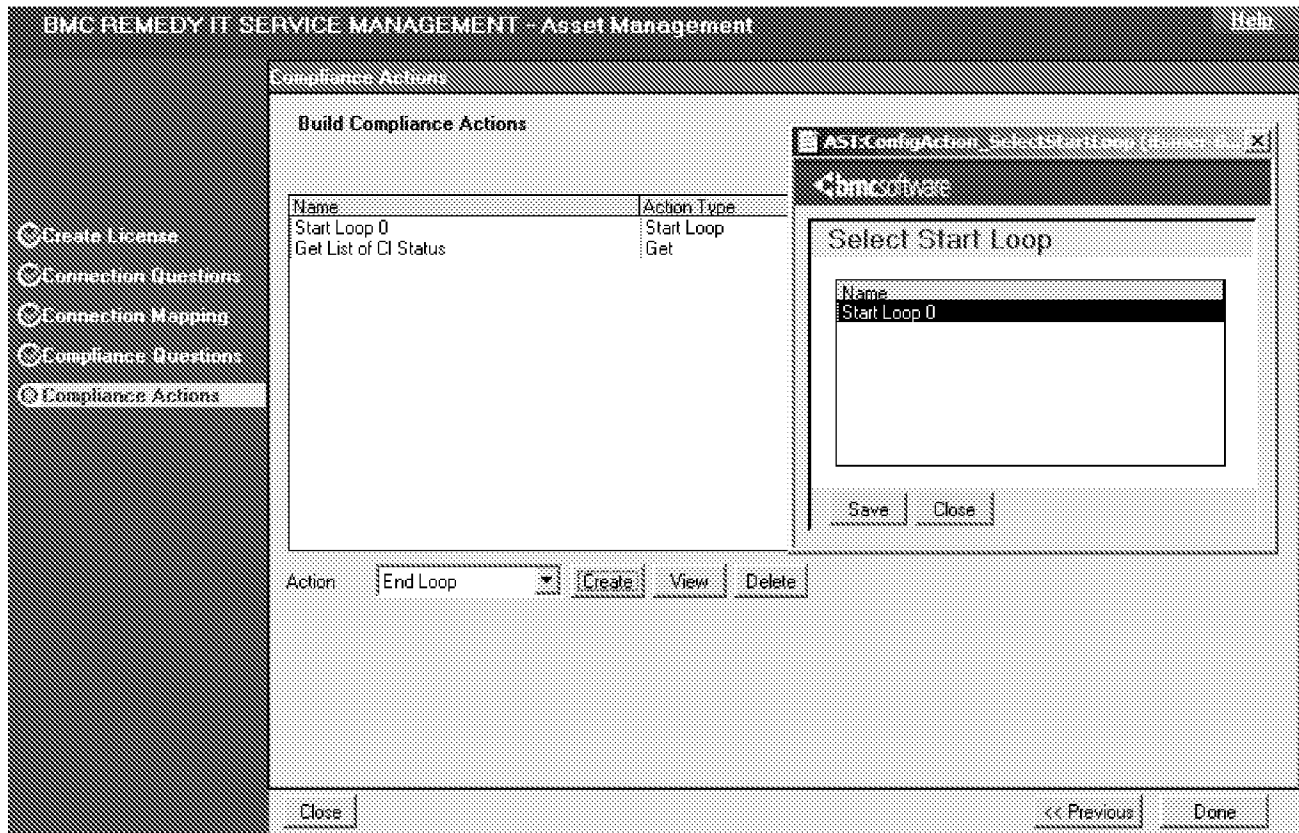
8 Specify compliance actions.

Based on the run-time data from the license engine and based on the compliance questions, select and create the following compliance actions:

- **Get** retrieves a list of values or a count of the matching records from the specified form. For more information, see “Get Compliance Action” on page 436.
- **Calculate** computes an expression. For more information, see “Calculate Compliance Action” on page 439.
- **Compare** performs a comparison defined in an expression. For more information, see “Compare Compliance Action” on page 440.
- **Update** updates a list of values on a specified form. For more information, see “Update Compliance Action” on page 441.
- **Start loop**—Lets you process a set of actions before moving on to the next. The Start Loop action is used to start a loop to return a list of values from a Get Action. A Start Loop Action must be created before the Get Action is created. Actions that use the Get action’s results list are created after the Get.

- **End loop**—Ends the loop before moving to the actions outside the loop. When you select End Loop, a pop up appears prompting you to select the Start Loop action for which the End Loop is to be created. Figure 15-17 shows an End Loop action being created.

Figure 15-17: End Loop being created for a Get action list



- 9 When you have finished specifying compliance actions, click Done.

Get Compliance Action

Use this action to retrieve information from BMC Remedy AR System or BMC Atrium CMDB forms.

► To create a Get compliance action

NOTE

This procedure continues from step 8 on page 435 and step 10 on page 447.

- 1 On the Compliance Actions page of the license type wizard, select Get in the Action field and click Create.

The Get Action dialog box is displayed as shown in Figure 15-18.

Figure 15-18: Get Action dialog box

The screenshot shows the 'Get Action' dialog box in BMC Software. The dialog is titled 'Get Action' and has a header bar with the BMC Software logo. The main content area contains the following fields and controls:

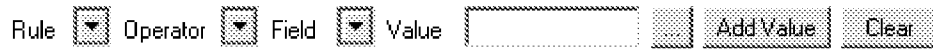
- Name:** A text field containing 'Get AR System License Types'.
- Source:** A dropdown menu set to 'AR'.
- From:** A dropdown menu set to 'AR System Licenses'.
- Type:** Radio buttons for 'List' (selected) and 'Count'.
- Select:** A dropdown menu set to 'License Type', with 'Add' and 'Remove' buttons.
- SELECT:** A large empty text area for entering SQL-like queries.
- Rule, Operator, Field, Value:** A row of dropdown menus and text fields for defining query criteria, with 'Add Value' and 'Clear' buttons.
- WHERE:** A text field for entering WHERE clauses.
- Buttons:** 'Save' and 'Close' buttons at the bottom.

- 2 Enter a name for the Get compliance action.
- 3 Select a source:
 - **CMDB** is used to get information from BMC Atrium CMDB forms based on the values of class attributes.
 - **AR** is used to get data from any BMC Remedy AR System form.
- 4 If the source is CMDB, a list of classes shown in the CI Type field and an attribute. (Classes are shown here instead of form names.) If you selected AR, select the form name from the From field.
- 5 Select the Get action type for which you want to create a query:
 - **List** returns a list of records that match the criteria specified in the Get action that you are building.
 - **Count** sums the number of records that match the criteria specified in the Get action that you are building.
- 6 In the Select field, select from the list of fields displayed for the form or class you selected, and click Add. If you selected Count in the previous step, this field is unavailable.

- 7 Build the WHERE qualification using the controls in the toolbar above the WHERE field.

As you build the WHERE qualification, it is displayed in the WHERE field shown in Figure 15-19.

Figure 15-19: WHERE qualifier toolbar



TIP

As you build the WHERE qualification, you can delete it to make corrections at any time by clicking Clear.

You can use all or some of the following fields to build your WHERE qualification.

- a Select the rule:
 - **GetCertificates** specifies the instance ID of the certificate as a value in the WHERE qualification.
 - **GetQuestions** specifies the compliance question that you select as a value in the WHERE qualification.
 - GetCertificates and GetQuestions are the default rules that appear. As you create actions, they also appear for you to select.

NOTE

Some of the data that is created through connection rules can also be selected here. For example, you can query relationship tables that were populated during a previous connection run.

- b Select the operator.
- c Select a field to enter as a value in the WHERE qualification.
- d To specify a free-form value, enter it in the Value field and click Add Value.
- 8 Click Save and then click Close.
- 9 After you have finished adding all of the compliance actions that you require, click Done to complete the license type wizard.

Calculate Compliance Action

Use the calculate compliance action to compute a value from data retrieved from previous actions. This computation can then be used to update other fields or storage values.

► **To create a Calculate compliance action**

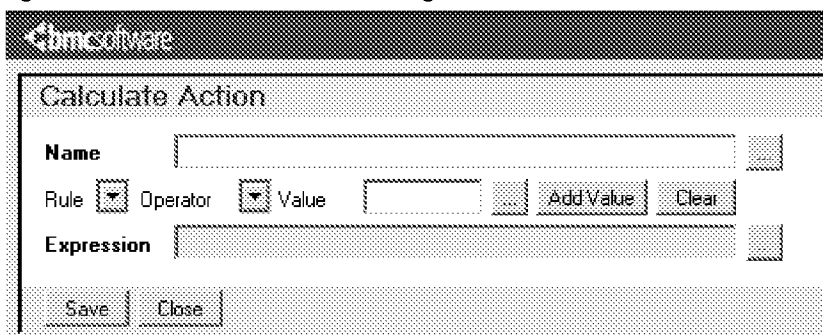
..... **NOTE**

This procedure continues from step 8 on page 435 and step 10 on page 447.

- 1 On the Compliance Actions page of the license type wizard, select Calculate in the Action field and click Create.

The Calculate Action dialog box is displayed as shown in Figure 15-20.

Figure 15-20: Calculate Action dialog box



- 2 Type a name for the calculate compliance action in the Name field.
- 3 Build the expression using the controls in the toolbar above the Expression field. As you build the expression, it is displayed in the Expression field shown in Figure 15-21.

Figure 15-21: Expression toolbar



..... **TIP**

As you build the expression, you can delete it to make corrections at any time by clicking Clear.

You can use the following fields to build your expression.

- a Select the rule:
 - **GetCertificates** specifies the instance ID of the certificate as a value in the expression.
 - **GetQuestions** specifies the compliance question you select as a value in the expression.
 - GetCertificates and GetQuestions are the default rules that appear. As you create actions, they also appear for you to select.

NOTE

Some of the data created due to connection rules can also be selected here. For example, you can query relationship tables that were populated during a previous connection run.

- b Select the operator.
 - c Specify a free-form value in the Value field and click Add Value.
- 4 Click Save and then click Close.
 - 5 After you have finished adding all of the compliance actions you require, click Done to complete the license type wizard.

Compare Compliance Action

Use the compare compliance action to compare two or more values and take action based on that comparison.

► **To create a Compare compliance action**

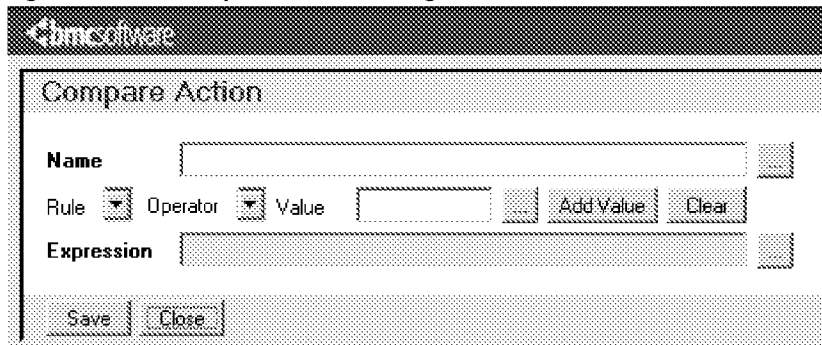
NOTE

This procedure continues from step 8 on page 435 and step 10 on page 447.

- 1 On the Compliance Actions page of the license type wizard, select Compare in the Action field and click Create.

The Compare Action dialog box is displayed as shown in Figure 15-22.

Figure 15-22: Compare Action dialog box



- 2 Type a name for the action in the Name field.
- 3 Build the expression using the controls in the toolbar above the Expression field. As you build the expression, it is displayed in the Expression field.

TIP

As you build the expression, you can delete it to make corrections at any time by clicking Clear.

You can use the following rules to build your expression.

- a Select the rule:

-
- **GetCertificates** specifies the instance ID of the certificate as a value in the expression.
 - **GetQuestions** specifies the compliance question you select as a value in the expression.
 - **GetCertificates** and **GetQuestions** are the default rules that appear. As you create actions, they also appear for you to select.

..... **NOTE**

Some of the data created due to connection rules can also be selected here. For example, you can query relationship tables that were populated during a previous connection run.

.....

- b Select the operator.
 - c Specify a free-form value in the Value field and click Add Value.
- 4 Click Save and then click Close.
 - 5 After you have finished adding all of the compliance actions you require, click Done to complete the license type wizard.

Update Compliance Action

Use the update compliance action to update fields on BMC Remedy AR System compliance forms or update compliance questions that are set to computed.

► To create an Update compliance action

..... **NOTE**

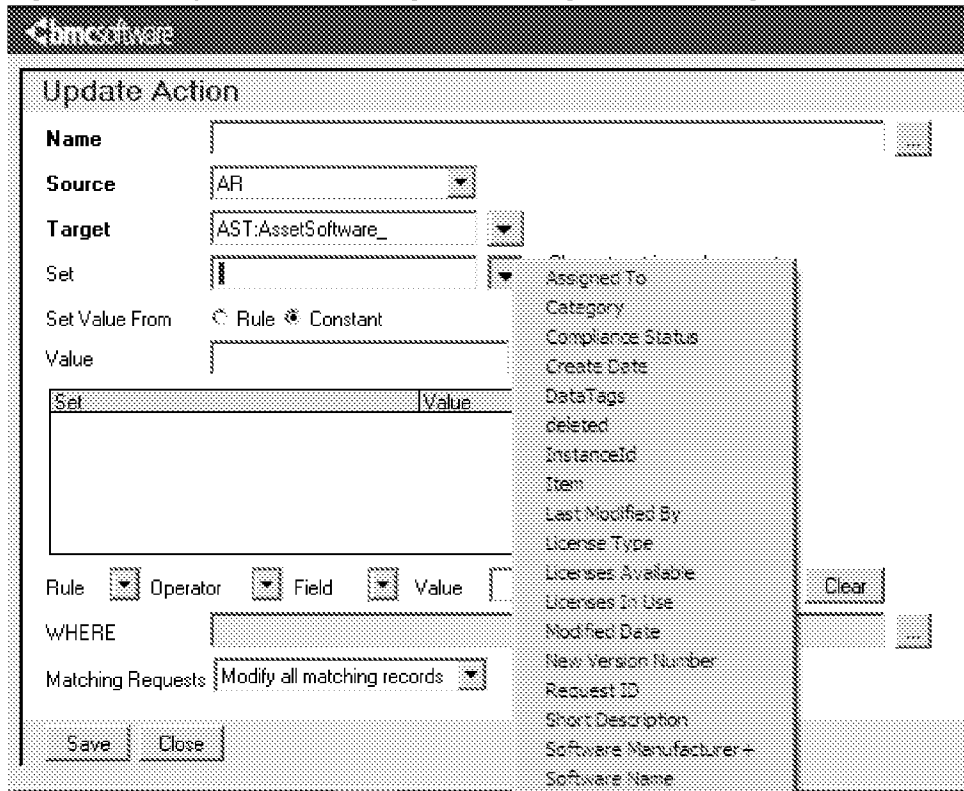
This procedure continues from step 8 on page 435 and step 10 on page 447.

.....

- 1 On the Compliance Actions page of the license type wizard, select Update in the Action field and click Create.

The Update Action dialog box is displayed as shown in Figure 15-23.

Figure 15-23: Update Action dialog box showing Set field listing fields in selected form



- 2 Type a name for the action in the Name field.
- 3 Select a source:
 - **AR** allows an AR form to be selected and fields on that form to be updated.
 - **Compliance Question** allows configured compliance questions to be selected. Only those that are set to computed are available for selection. In addition to these questions, three compliance flags can also be set:
 - **Compliance flag** is set to true (value equal to 1) or false (value equal to 0) to indicate if certificates are in or out of compliance.
 - Breach levels can be set by an update rule to indicate that a certificate is approaching or about to be in breach compliance. Two breach flags are available for selection: **Breach level 1** and **Breach level 2**. As an example, you would set Breach level 1 to warn you when the number of licenses you have was reduced to 10; you would set Breach level 2 to warn you when the number of licenses you have was reduced to 2. Breach level 3 is not available for selection because it is set automatically by workflow when the license certificate goes out of compliance.
- 4 In the Target field, select a form name that contains the fields that will be updated. AST:Compliance_BasicQuestions is the compliance form for the Basic model and AST:Compliance_AdvancedQuestions is the compliance form for the Advanced model.

-
- 5 In the Set field, select the field to be updated.
 - 6 In the Set Value From field, select the method to use for the update:
 - **Rule** uses the rule (WHERE qualifier) that you build in the WHERE field. If you select Rule, continue with step 7.
 - **Constant** uses the value you specify in the Value field (just below the Set Value From field). If you select Constant, skip to step 8.
 - 7 If you selected Rule in step 6, build the WHERE qualifier using the controls in the toolbar above the WHERE field. As you build the clause, it is displayed in the WHERE field.

..... **NOTE**

If you selected Compliance Question for Source, the following default WHERE qualifier is created. You can modify this WHERE qualifier as required.
"GetCertificates.InstanceId" = 'Certificate_InstanceID'

..... **TIP**

As you build the WHERE qualifier, you can delete it to make corrections at any time by clicking Clear.

You can use the following rules to build your expression.

- a Select the rule:
 - **GetCertificates** specifies the instance ID of the certificate as a value in the WHERE qualifier.
 - **GetQuestions** specifies the compliance question you select as a value in the WHERE qualifier.
 - GetCertificates and GetQuestions are the default rules that appear. As you create actions, they also appear for you to select.

..... **NOTE**

Some of the data created due to connection rules can also be selected here. For example, you can query relationship tables that were populated during a previous connection run.

- b Select the operator.
 - c Select a field to enter as a value in the WHERE qualifier.
 - d To specify a free-form value, enter it in the Value field and click Add Value.
- 8 If you selected Constant in step 6, enter the value in the Value field below the Constant option and click Add.

..... **TIP**

You can delete values that you have entered by selecting them from the list and clicking Remove.

- 9 Select a matching requests value:
 - **Do not update, issue an error** —Issue an error when the first matching record is encountered and abort the run.
 - **Do not update, issue a warning** —Issue a warning for each matching record and continue until all records have been processed and the run is completed.
 - **Modify all matching records** —Update all records that match.
- 10 Click Save and then click Close.
- 11 After you have finished adding all of the compliance actions you require, click Done to complete the license type wizard.

Advanced mode

This section provides instructions for the following tasks:

- Customizing the connection and compliance template forms so that they can then be mapped using the license type wizard. To customize these template forms, follow the steps in “To customize your connection and compliance forms” on page 444.
- Completing the license type wizard after you select Advanced in the License Type Mode field of the License Type Information page. To complete the rest of the steps in the license type wizard after selected Advanced mode, follow the steps in “To continue creating a license type using advanced mode” on page 445.

► To customize your connection and compliance forms

- 1 Open the AST:ConnectionTemplate form in Developer Studio.
The form appears with default field types and labels.
- 2 Save the template form under a new name. You will enter this name in a field in the license type wizard.
- 3 Change the labels and field characteristics so that they can be understood by the individual who will be creating the license certificate that uses this customized form. This form will appear for the user to enter data in when creating a license certificate.

..... **NOTE**

The ID for each field has to be a unique value within the range 303000500 – 303000900.

.....

..... **TIP**

You can optionally add workflows or filters to make it easier to enter data, to validate data, or both.

.....

- 4 Save the form.
- 5 Open the AST:ComplianceTemplate form in BMC Remedy Developer Studio and complete step 1 through step 4 for it.

► To continue creating a license type using advanced mode

- 1 If you have not already done so, complete step 1 through step 4 in “To configure a new license type” on page 426. In step 4, select Advanced.

Two additional fields are displayed as shown in Figure 15-24, which are *not* displayed for Basic mode:

- Connection Form Name
- Compliance Form Name

Figure 15-24: License type wizard - Create License Type (advanced mode)

- 2 In the Connection Form Name field, enter the name of the connection form that you customized from the AST:ConnectionTemplate form to create this license type. See the steps in “To customize your connection and compliance forms” on page 444 for more information about customizing this form.

IMPORTANT

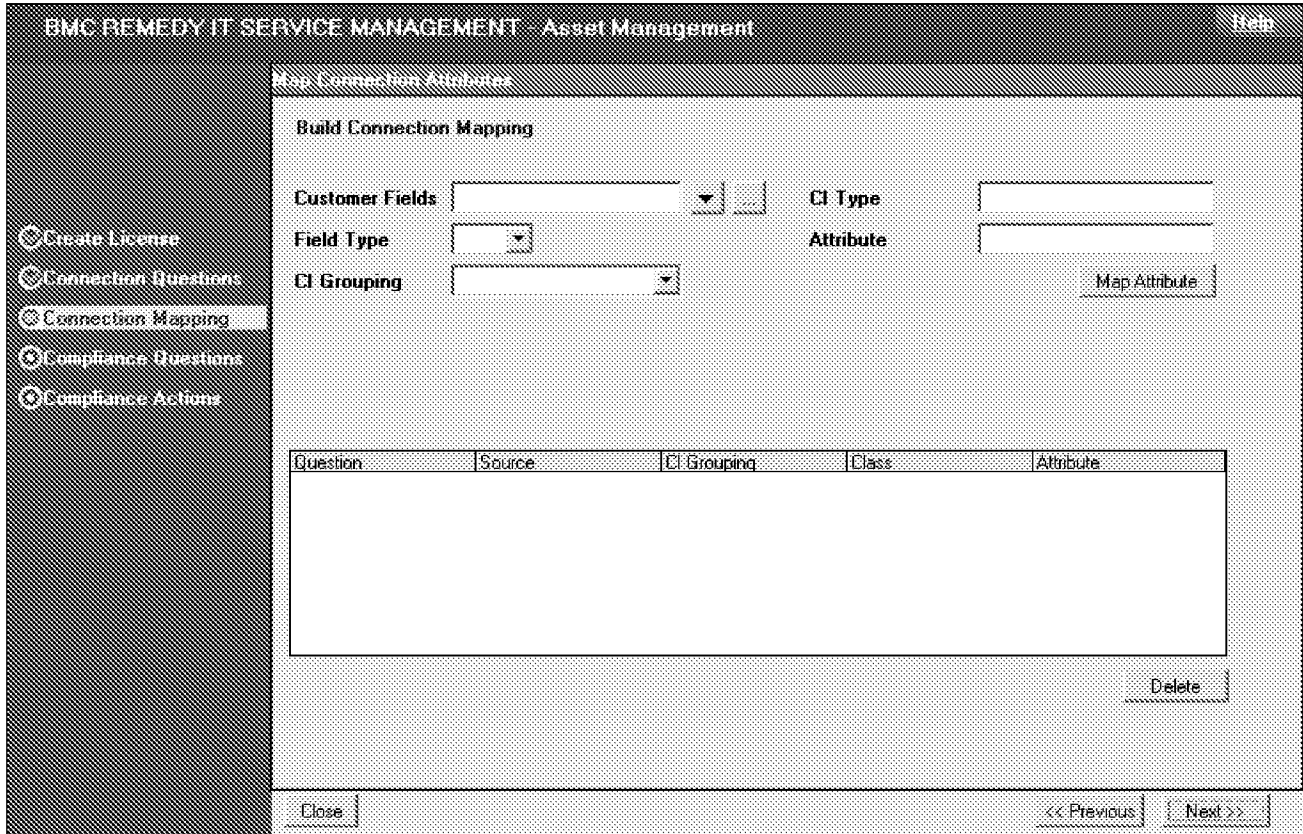
You must spell your form names correctly when you enter them. Otherwise, the license type wizard will not be able to retrieve the fields for selection.

- 3 In the Compliance Form Name field, enter the name of the compliance form that you customized from the AST:ComplianceTemplate form to create this license type. See the steps in “To customize your connection and compliance forms” on page 444 for more information about customizing this form.

4 Click Next.

The Map Connection Attributes page of the license type wizard for advanced mode appears as shown in Figure 15-25.

Figure 15-25: License type wizard - Map Connection Attributes (advanced mode)



- 5 From the Customer Fields list, select a field from the connection form that you customized.
- 6 Select a field type of either Char or Int.
The field type determines whether the user enters a character or integer answer to the question.
- 7 Select the CI grouping for the license type.
When you select a CI grouping, the certificates created with this type are grouped under a master certificate, based on license type and the product categorization you select in the CI Grouping field.
- 8 Specify how to map the questions to CIs in BMC Atrium CMDB.
For each connection question, select the CI Type and Attribute and click Map Attribute.

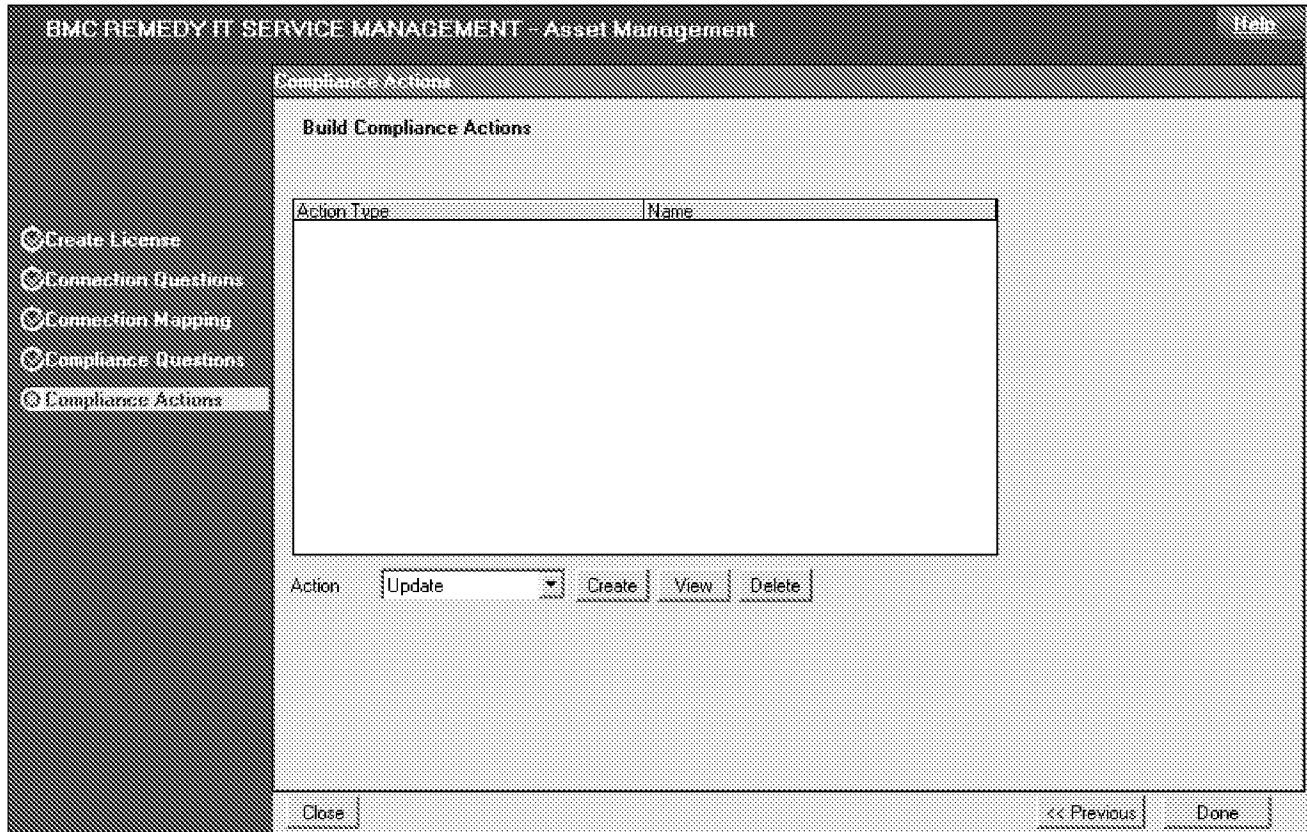
TIP

To delete a mapped attribute, select it from the list at the bottom of the Build Connection Mapping list and click Remove.

9 Click Next.

The Compliance Actions page of the license type wizard in advanced mode appears as shown in Figure 15-26.

Figure 15-26: License type wizard - Compliance Actions (advanced mode)



10 Specify compliance actions.

Based on the run-time data from the rules engine and based on the compliance questions, select and create the following compliance actions:

- **Get** retrieves a list of values or a count of the matching records from the specified form. For more information, see “Get Compliance Action” on page 436.
- **Calculate** computes an expression. For more information, see “Calculate Compliance Action” on page 439.
- **Compare** does a comparison based on an expression. For more information, see “Compare Compliance Action” on page 440.
- **Update** modifies a list of values on a specified form. For more information, see “Update Compliance Action” on page 441.
- **Start loop** processes a set of actions before moving on to the next. The Start Loop action is used to start a loop to return a list of values from a Get Action. A Start Loop Action must be created before the Get Action is created. Actions that use the Get action’s results list are created after the Get.

- **End loop** ends the loop before moving to the actions outside the loop. When you select End Loop, a pop up appears prompting you to select the Start Loop action for which the End Loop is to be created. Figure 15-17 shows an End Loop action being created.

11 When you have finished specifying compliance actions, click Done.

Enabling and disabling license types

When you create a license type as described in “Configure License Type” on page 417, it is created in draft mode. To use the license type, you must enable it.

..... **NOTE**

All license types have to enabled, including the license types shipped with the BMC Remedy Asset Management application. For more information about the license types shipped with the BMC Remedy Asset Management application, see “Default system license types” on page 418.

.....

► **To enable a license type**

- 1 From the Application Administration Console, click the Custom Configuration tab.
- 2 From the Application Settings list, choose Asset Management > Advanced Options > Configure License Type, and then click Open.

The Configure Licenses form lists the current license types.

- 3 From the list, select the license type that you want to enable and click Enable.

..... **TIP**

To prevent certain license types from being used, you can disable them by selecting them from the list and clicking Disable.

.....

..... **TIP**

You can continue learning more about the different types of questions you can ask and the mappings and actions you can use by looking at the default license types shipped with the BMC Remedy Asset Management application. Select one of the default license types from the Configure License Type dialog box that appears when you double-click Configure License Type on the Custom Configuration tab of the Application Administration Console, and click View. For more information about the default license types, see “Default system license types” on page 418.

.....

Troubleshooting license types

This section describes how to troubleshoot license types and the various rules and operations that they use.

When you use the license type wizard to create a license type, a set of rules is generated in the License Engine forms and stored as metadata. These forms start with a prefix of RLE. When the license engine runs, it executes these rules.

If your license job is not running properly, some of the reasons might be:

- The rules might not have been set up as required.
- The right qualifications might not have been defined.
- The fields might not have been mapped correctly.
- The right scope for the job might not be defined.

Make sure to test your new customized license type before you put it into production.

If everything has been set up correctly and the right scope has been defined for the license job, you should follow the debugging suggestions in “Logging” on page 450 and “Debugging tips” on page 450.

..... **NOTE**

Before trying to debug licenses, familiarize yourself with the rules engine described in the *BMC Remedy IT Service Management (ITSM) Architecture White Paper*.

.....

..... **IMPORTANT**

Before you begin debugging, export the license type forms as a backup. For a list of the forms you need to export and the method to follow, see the next section, “Backing up the license type forms”.

.....

Backing up the license type forms

To ensure there is no loss of data that prevents you from using your existing license types, before you begin troubleshooting, export the license type forms listed in Table 15-2 using BMC Remedy Developer Studio. Then you can import them at a later time if you need to recover them.

Table 15-2: List of license type forms

Form name
AST:ConfigAction_CalculateSpecificData
AST:ConfigAction_GetSpecificData
AST:Config_ActionBase
AST:ConfigCompliance_RuleReference
AST:ConfigConnection_RuleFieldMapping
AST:ConfigCompliance_RuleDefinition
AST:ConfigBasicQuestionRegistry_DispProp
AST:ConfigBasicQuestionRegistry_Base
AST:ConfigLicenseTypeRegistry_DispProp
RLE:ComplexRuleSpecificData
RLE:CalculationRuleSpecificData
RLE:GetRuleSpecificData
RLE:LoopRuleSpecificData
AST:ConfigLicenseTypeRegistry_Base

Table 15-2: List of license type forms (Continued)

Form name
AST:ConfigCompliance_ActionSelectList
AST:ConfigAction_UpdateSpecificData
AST:ConfigRuleSet
RLE:RuleSet
RLE:RuleSetType
RLE:BaseRule
RLE:UpdateRuleSpecificData

Logging

To help debug problems, you can turn logging on at the engine level. To log all of the rules as they execute, you can set the log level to debug. Then you can examine the rules to determine whether they are being translated to the right fields, are returning the right results, and so on. The log file is in the DB directory of the BMC Remedy AR System server and is called `arjavaplugin.log`. For more information about setting the log level, see “License Engine Configuration” on page 455.

Debugging tips

Every License type created by the license type wizard creates a corresponding set of rules that are stored in the license engine rule forms and start with a prefix of `RLE:.` The corresponding rules created are:

- Query operations translate into GET rules.
- Create/modify operations translate into UPDATE rules.
- Computational operations translate into CALCULATE rules.
- Comparison operations translate into COMPARE rules.

To debug license types, try some of these steps in the procedures in:

- “If a license type is not running” on page 451
- “If the right CIs are not connected to a certificate” on page 451
- “If compliance actions are not working correctly” on page 452.

..... IMPORTANT

Make sure not to delete the data in the forms listed below directly because the functionality for the corresponding license types will not work. Also there are a few system rules shipped with 7.5.00 that are loaded in the RLE forms at installation time. These rules should not be modified or deleted since they are used by all license types.

The following is a list of some of the RLE forms and the system rules they contain:

- ❖ The RLE:RunTag form has a system rule for the LIVE RunTag
- ❖ The RLE:RuleSetType form has two system data rules: one for CONNECTION jobs and the other one for COMPLIANCE jobs
- ❖ The RLE:RuleSet, RLE:GetRuleJoin, RLE:UpdateRuleJoin, and RLE:LoopRuleJoin forms have two rulesets called GetScope and CommitData. These are used to process connection rules for all rules.
- ❖ Several other RLE forms are also installed with all the default license types that are shipped with the BMC Remedy Asset Management application. Do not modify any of this data in the forms. Only modify or delete license types using the license type wizard.

▶ **If a license type is not running**

- 1 Make sure that the license type is enabled as described in “Enabling and disabling license types” on page 448.

▶ **If the right CIs are not connected to a certificate**

- 1 Examine your connection rules to ensure the following points:
 - ❖ If you are using the basic model, you have the right questions mapped to the CMDB classes/fields.
 - ❖ If you are using the advanced model, you have the right fields mapped to the CMDB fields.
- 2 The following is the sequence of operations that occur when a connection job is run. You can track the sequence of these operations in the `arjavaplugin.log` if needed by turning debugging on as described in “License Engine Configuration” on page 455.
 - a A scope list is updated in a backend table called `AST:LicenseScopeTable`, based on the `qual` parameter defined for the job.
 - b The system gets the certificates for the license type.
 - c For every certificate, the system gets the matching CIs.
 - d The system updates the scope list with the CIs that have matching certificates.
 - e The system performs further operations on the scope table, for example, CIs get related to certificates, exception data gets processed, and so on.

► **If compliance actions are not working correctly**

- 1 Because the results of previous actions can be used in subsequent actions, ensure that all of the actions are in the right sequence.
- 2 Ensure that each action has the correct WHERE qualifiers, so that the right set of data is returned.

Create Contract type

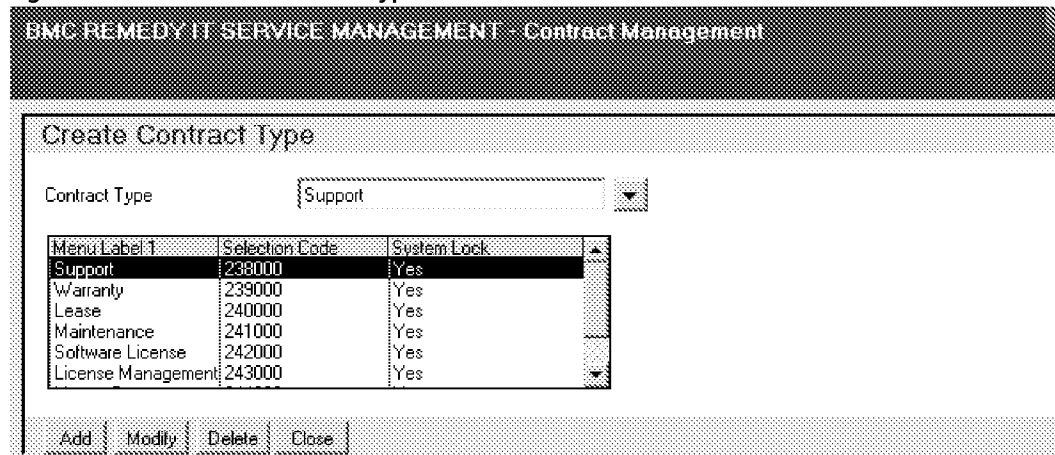
If you need contract types beyond those that come with the BMC Remedy Asset Management application, you can create your own contract types. These contract types are available when a contract manager creates a new contract on the Contract Management console. For more information, see the *BMC Remedy Asset Management User's Guide*. An example of a typical contract type that you might create is Statement of Work.

► **To create a contract type**

- 1 From the Application Administration Console, select the Custom Configuration tab.
- 2 From the Application Settings list, choose Asset Management > Advanced Options > Create Contract Type, and then click Open.

The Create Contract Type form is displayed as shown in Figure 15-27.

Figure 15-27: Create Contract Type form



- 3 On the Create Contract Type form, type the name of the contract type and click Add.

A BMC Remedy AR System User Note message window indicates that the contract type has been created.

TIP

To modify or delete a contract type, select it from the list and click Modify or Delete.

Configure Inbox Preferences



The inbox is used to track events that individuals need to take care of, such as:

- Contracts that have expired or are about to expire.
- Software license certificates that are not compliant or are approaching noncompliance.
- Software license certificates that have expired or are about to expire.
- Summary results of the engine run.
- List of new CIs that are linked to software license certificates.
- A rule from the Job History form that does not complete successfully.
- Events from the exception table, which can be pushed to the inbox.

From the inbox, you can indicate whether you have dealt with a message. To view the applicable contract or certificate, select the message and click View.

This section describes how to set up preferences for the inbox. For more information about using the inbox, see the *BMC Remedy Asset Management User's Guide*.

► To configure inbox preferences

- 1 From the Application Administration Console, click the Custom Configuration tab.
- 2 From the Application Settings list, choose Asset Management > Advanced Options > Inbox Preferences, and then click Open.

The Inbox Preferences form is displayed as shown in Figure 15-28.

Figure 15-28: Inbox Preferences form

- 3 Select the request type. All of the request types for contracts, certificates, and jobs are available for selection.

When you select a request type, the Event Type field is populated with values. Table describes the event types that are available based on the request type that you select.

Table 15-3: Resulting event types based on request type selected

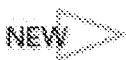
Request type value	Corresponding event types
※ Lease	Expiring
※ Maintenance	Expired
※ Master Contract	
※ Support	
※ Warranty	
License Management	Certificates Connected with CIs
Exceptions	Certificates Out of Compliance
	CIs with Multiple Certificates
	CIs without Certificates

Table 15-3: Resulting event types based on request type selected (Continued)

Request type value	Corresponding event types
License Certificate or Software License	Expiring
	Expired
	Not Compliant

- 4 Based on the request type that you selected, select an event type.
- 5 Select an importance:
 - ❖ **High** specifies the highest level of importance.
 - ❖ **Medium** specifies a middle level of importance.
 - ❖ **Low** specifies the lowest level of importance.
- 6 Select an attention flag:
 - ❖ **Needs attention** specifies that the contract or license certificate requires that action be taken. This is the default status when a license certificate or contract is created.
 - ❖ **Dealt with** indicates a status of Dealt with, which is set from the More Details dialog box that is displayed from a license certificate or contract.
 - ❖ **Informational** specifies that no action is required.
- 7 Select a view access:
 - ❖ **Public** specifies that events can be seen by everyone who belongs to the specified company.
 - ❖ **Group** specifies that events can be seen only by individuals in the specified support group.
 - ❖ **Individual** specifies that events can be seen only by individuals within the specified support group.
- 8 Select a company.
- 9 Select a status.
- 10 Fill in the support company, organization, and group fields.
- 11 Click Save and then click Close.

License Engine Configuration



The license engine connects CIs to software licenses and checks compliance of certificates. The license engine needs to be run to process license types and actions that you configure as described in “Configure License Type” on page 417. There are certain parameters you can configure that control:

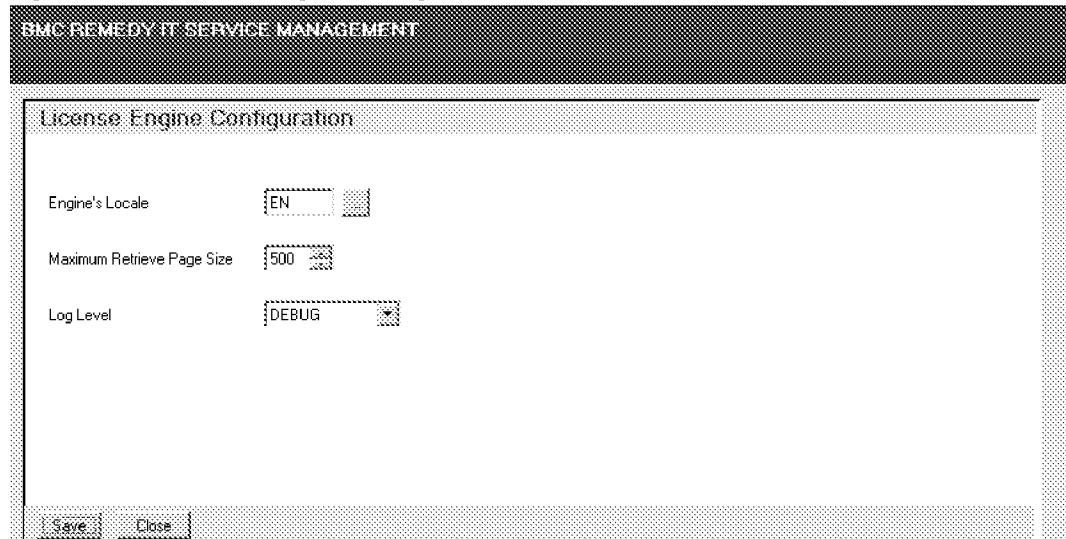
- ❖ The language in which the rules engine runs
- ❖ Memory retrieval page size
- ❖ Type and level of information captured in the log file.

► **To configure the license engine**

- 1 From the Application Administration Console, click the Custom Configuration tab.
- 2 From the Application Settings list, choose Asset Management > Advanced Options > License Engine Configuration, and then click Open.

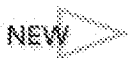
The License Engine Configuration form is displayed as shown in Figure 15-29.

Figure 15-29: License Engine Configuration form



- 3 Enter the language in which the engine is to run. This determines the language used by the exception table. The default is EN for English.
- 4 Select the maximum page retrieval size for memory swapping. This parameter can help control the performance of your system.
- 5 Select the type and level of information you want to be written to the log files when the engine runs. By default, this is set to INFO and if you are in the process of debugging then at that time you would set it to DEBUG. Setting it DEBUG can degrade performance.
- 6 Click Save and then click Close.
- 7 After you make changes to these settings, you have to restart your server for the changes to take effect.

Rules



You use the Configure CI Rules form to configure the method of assignment used to automatically select an assignee (not the assigned support group, but the individual within the group) for purchase requisitions in the BMC Remedy Asset Management application. Each company defined in the Company form can have its own assignment rules.

You can configure assignment for one company, or you can define it for all companies using the Global option. When the BMC Remedy Asset Management application uses assignment, for example, when assigning a purchase requisition for assignment, the BMC Remedy Assignment Engine checks the assignment rules for the requester's company. If no rules are found, it uses the rules defined for Global.

You can also use this form to set up preferences for automatically running the engine. For more information see "Configure License Engine Run" on page 458.

► **To configure assignment rules for the BMC Remedy Asset Management application**

- 1 From the Application Administration Console, click the Custom Configuration tab.
- 2 From the Application Settings list, choose Asset Management > Advanced Options > Rules, and then click Open.

The Configure CI Rules for is displayed as shown in Figure 15-30.

Figure 15-30: Configure CI Rules form

The screenshot shows the 'Configure CI Rules' form in the BMC Remedy IT Service Management - Asset Management application. The form has a header bar with the application name and a 'Help' link. Below the header, the title 'Configure CI Rules' is displayed. The form contains several fields and controls:

- Company***: A dropdown menu with 'Calbro Services' selected.
- Assignment**: A tabbed interface with 'License Engine' selected.
- Assignment Engine Integration**: A section with a radio button for 'Yes' (selected) and a radio button for 'No'. To the right is a 'Description' text area.
- Assignment Process**: A dropdown menu with 'Asset General Assignment - Number' selected.
- Rule ID**: A text input field containing 'ARU'.

- 3 Select the Assignment tab.
- 4 On the Configure CI Rules form, from the Company list, select the company for which you want to configure assignment, or select the Global option.
Selecting Global configures assignment for all companies.
- 5 To enable assignment, select Yes for Assignment Engine Integration.

- 6 From the Assignment Process list, select the default process for the assignment. The options are:

- **Asset General Assignment - Capacity**—Capacity in the BMC Remedy Assignment Engine is a ratio-based assignment method.

For example, you have two assignees, Assignee A and Assignee B. In the People form, there is a field to set up a capacity number for a person. You can use any number. For example, if you assign Assignee A the number 100 and Assignee B the number 200, Assignee B can handle twice as many assignments as Assignee A. The BMC Remedy Assignment Engine assigns records (for example, purchase requisitions or tickets) to Assignee A every time two records have been assigned to Assignee B.

Specify the capacity for each support staff person in the Capacity Rating field on the people form. See “Adding a support staff person” on page 117.

- **Asset General Assignment - Number**—Tracks the number of tickets assigned to the person. The number assignment process selects the person with the least number of tickets already assigned.
- **Asset General Assignment - Round Robin**—Tracks the last time the person received an assignment. The round robin assignment process selects the person who was least recently assigned an incident.

- 7 Optionally, enter a description.
- 8 For Status, select Enabled.
- 9 Click Save and then click Close.

Configure License Engine Run

This section describes how to configure the license engine to run when you want it to and to perform the functions you want when it is run.

The license engine needs to be run to:

- Associate orphaned CIs with contracts
- Remove expired CIs from the system.

► To run the license engine

- 1 From the Application Administration Console, click the Custom Configuration tab.
- 2 From the Application Settings list, choose Asset Management > Advanced Options > Rules, and then click Open.

- 3 Select the License Engine tab as shown in Figure 15-31.

Figure 15-31: License Engine tab of the Configure CI Rules form

The screenshot shows the 'Configure CI Rules' form in BMC Remedy IT Service Management. The form is titled 'Configure CI Rules' and is for the 'License Engine' tab. The 'Company' field is set to 'Calbro Services'. The 'Assignment' is 'License Engine'. The 'Run Tag Name' field is empty. The 'Drop CIs after License Expiry' field has a 'Yes' radio button selected. The 'Days before drop' field is empty. The 'Run Engine?' field has a 'Yes' radio button selected. The 'Run Engine on CI Update' field is set to 'No'. The 'Prompt Engine Run' field is set to 'No'. There are three informational text blocks on the right side of the form: 'Drop CIs after License Expiry: - this drops the CI relationships after the number of days specified', 'Run Engine on CI Update: - on manual modification of a CI, prompts user to run engine - on reconciliation/automatic update of a CI, automatically runs engine.', and 'Prompt Engine Run: - on modification of certificate (includes modification to key attributes or adding or removing CI associations), prompts user to run engine.'

- 4 Select the LIVE run tag name.
- 5 Select the options to specify how and when CIs should be dropped:
 - a If you want to drop CIs after they have expired when the license engine is run, select Drop CIs after License Expiry.
 - b Now you can select number of days after their expiry date when the CIs should be dropped.
 - c From the Run Engine drop-down, select whether you want the license engine to run or not. The default is Yes.
When you select Yes, the engine will run against both connection and compliance rules.
- 6 Select whether or not you want the license engine to be triggered to run when changes or deletions have been made to CIs and certificates.
- 7 Click Save and then click Close.

Sync Asset UI with CMDB

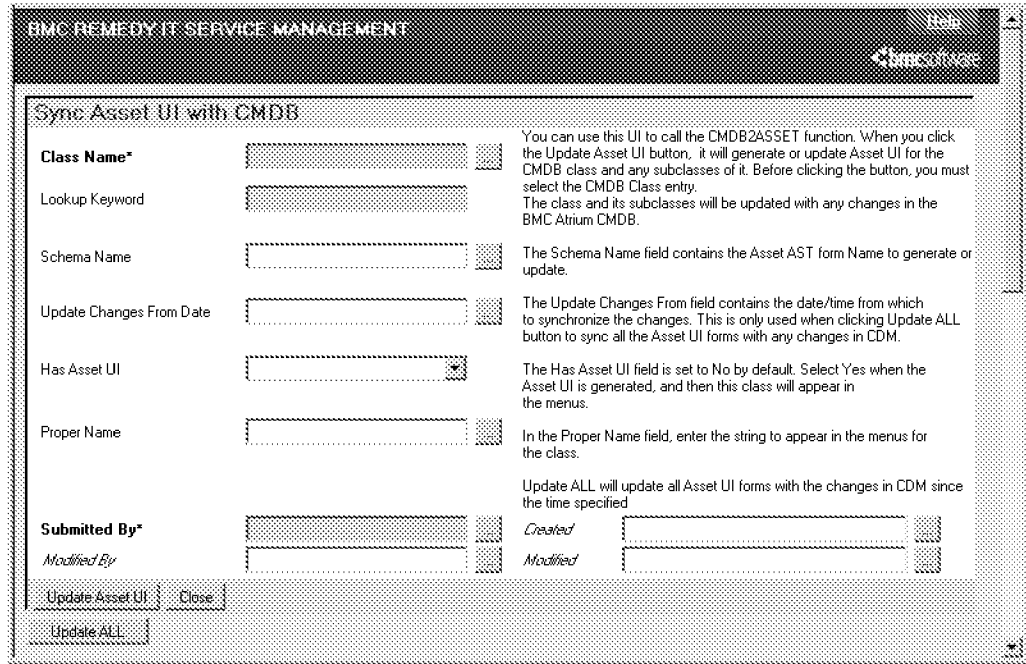
Use this feature to synchronize the BMC Remedy Asset Management forms with BMC Atrium CMDB.

► To synchronize Asset Management forms with BMC Atrium CMDB

- 1 From the Application Administration Console, click the Custom Configuration tab.
- 2 From the Application Settings list, choose Asset Management > Advanced Options > Sync Asset UI with CMDB, and then click Open.

The Synch Asset UI with CMDB form appears as shown in Figure 15-32.

Figure 15-32: Sync Asset UI with CMDB form



- 3 On the Sync Asset UI with CMDB form, click Search, and then select the BMC Remedy Asset Management form to synchronize with BMC Atrium CMDB.
- 4 To create an BMC Remedy Asset Management form, enter the form name in the Schema Name field.
- 5 In the Update Changes From Date field, optionally enter the starting date and time for the changes to be updated. Leave blank to include any changes that occurred after the last update.
- 6 To make the class available in the BMC Remedy Asset Management application, in the Has Asset UI field, select Yes.
- 7 In the Proper Name field, enter the name to appear in BMC Remedy Asset Management forms.
- 8 Click Update Asset UI or Update ALL.

Unavailability Priority

Use this feature to configure the priority of a CI unavailability record based on the unavailability class (Incident or Change) and type (scheduled or unscheduled).

The Company field on this form is disabled to allow for a global priority mapping only.

► To configure unavailability priority

- 1 From the Application Administration Console, click the Custom Configuration tab.
- 2 From the Application Settings list, choose Asset Management > Advanced Options > Unavailability Priority, and then click Open.

The Configure CI Unavailability Priority form appears as shown in Figure 15-33.

Figure 15-33: Configure CI Unavailability Priority form

The screenshot shows the 'Configure CI Unavailability Priority' form in the BMC Remedy IT Service Management interface. The form is titled 'Configure CI Unavailability Priority' and is located within the 'Asset Management' section. The form contains several fields and a status section:

- Company***: Subal
- Unavailability Class***: Change
- Unavailability Type***: Scheduled Full
- Priority***: Medium
- Submitter***: mmanager
- Submit Date**: [Empty field]
- Last Modified By**: [Empty field]
- Last Modified Date**: [Empty field]
- Unavailability Priority ID**: AUP
- Status***:
 - Proposed
 - Enabled
 - Offline
 - Obsolete
 - Archive
 - Delete

At the bottom of the form, there are 'Save' and 'Close' buttons.

- 3 On the Configure CI Unavailability Priority form, from the Unavailability Class list, select Incident or Change.
- 4 Select the unavailability type for which you are configuring the priority from the following options:
 - **Scheduled Full**—This indicates plans to take a CI out of service, typically during a scheduled change.
 - **Scheduled Partial**—This indicates plans to change the CI, but not to take it out of service. The CI performance will suffer some degradation during the duration of the change.
 - **Unscheduled Full**—The CI is experiencing a complete service outage that was not planned.
 - **Unscheduled Partial**—The CI is experiencing a service degradation that was not planned.
- 5 From the Priority list, select a priority.

- 6 For Status, select Enabled.
- 7 Click Save.

Approval Process for CI Configurations

This feature is used to configure the default criteria for approving configurations. If the BMC Remedy Change Management application is installed, the application creates a change request for the configuration and sends notification. Otherwise, the application sends notification.

For example, if you define the approval process for company A on this form, when you create a configuration from the Manage Configurations link in the BMC Remedy Asset Management application for company A, the application sends the request for approval to the group or individual specified to approve configurations for that company.

For more information, see Chapter 12, “Configuring approvals.”

► **To configure the approval process for configurations**

- 1 From the Application Administration Console, click the Custom Configuration tab.
- 2 From the Application Settings list, choose Asset Management > Approval > Approval Process for CI Configurations, and then click Open.

The Configure Approval Process for Configurations form appears as shown in Figure 15-34.

Figure 15-34: Configure Approval Process for Configurations form

The screenshot shows a web-based form titled "Configure Approval Process for Configurations" within the BMC Remedy IT Service Management - Asset Management application. The form is organized into several sections:

- Company Information:** Fields for Company (ABC Company), Region, Site Group, and Site, each with a dropdown menu and a "Clear" button.
- Change Template:** A dropdown menu with a "Clear" button. A note states: "If a Change Template is not chosen, then the timing of the Change, Impact, and Urgency must be selected".
- Change Timing, Impact, and Urgency:** Three dropdown menus with values "Normal", "4-Minor/Localized", and "3-Medium" respectively, each with a "Clear" button.
- Risk Level and Lead Time:** Two dropdown menus, each with a "Clear" button.
- Group Notifications:** Fields for Notification Company (ABC Company), Notification Organization (Customer Support), and Notification Group+ (Desktop Management), each with a dropdown menu and a "Clear" button.
- Individual Notifications:** Fields for Company, First Name, Last Name, and Notification Contact, each with a dropdown menu and a "Clear" button.

At the bottom of the form, there are "Search" and "Close" buttons.

-
- 3 On the Configure Approval Process for Configurations form, click the New Request toolbar icon in BMC Remedy User or the New button in your browser.
 - 4 From the Company field, select the company whose approval process you want to configure.
 - 5 To configure a separate approval process for a specific region, site group, or site, make the appropriate selections.
 - 6 From the Group Notifications and Individual Notifications lists, select the group or individual to notify to approve configurations.
 - 7 If the BMC Remedy Change Management application is installed, you must also configure the change request generated for a new configuration:
 - If there is an appropriate change template, select it. The change template indicates the change timing, impact, urgency, risk level, and lead time.
 - If you do not select a change template, you must select the change timing, impact, urgency, and risk level. Optionally, you can specify the lead time to indicate the number of hours of preparation required before a change can be scheduled for implementation.

NOTE

Risk Level 1 is the *lowest* risk and Risk Level 5 is the *highest*.

- 8 Click Save.

CMDB Class Manager Console

You can use the CMDB Class Manager Console to view and manage the data model. The classes you view represent CIs and relationships. To access this function, from the Application Administration Console, choose Asset Management > CMDB Class Manager Console > CMDB Class Manager Console, and then click Open. For detailed information about this console, see the *BMC Atrium CMDB Administrator's Guide*, the *BMC Atrium Core Installation Guide*, and the *BMC Atrium CMDB User's Guide*.

Asset Reconciliation Configuration Console

The BMC Remedy Asset Management application uses the Reconciliation Engine to:

- Reconcile the sandbox dataset against the production dataset if you configured a sandbox dataset, as described in "BMC Remedy Asset Management settings" on page 406.
- Delete CIs that you selected to be deleted, as described in "CI Deletion" on page 410.

You can configure the Reconciliation Engine from the Asset Administration Console. To access this function, choose Asset Management > Reconciliation Engine > Asset Reconciliation Configuration Console, and then click Open.

For details, see the *BMC Atrium Core Installation Guide* and the *BMC Atrium CMDB Administrator's Guide*.

B

Create license type examplesv

This section provides examples to show you how to create license types in the BMC Remedy Asset Management application. For more information about creating license types, see “Configure License Type” on page 417.

The following topics are provided:

- Site license type example—basic mode (page 498)
- Per instance license type example—basic mode (page 508)
- BMC Remedy AR System fixed and floating license type example—advanced mode (page 520)

Site license type example—basic mode

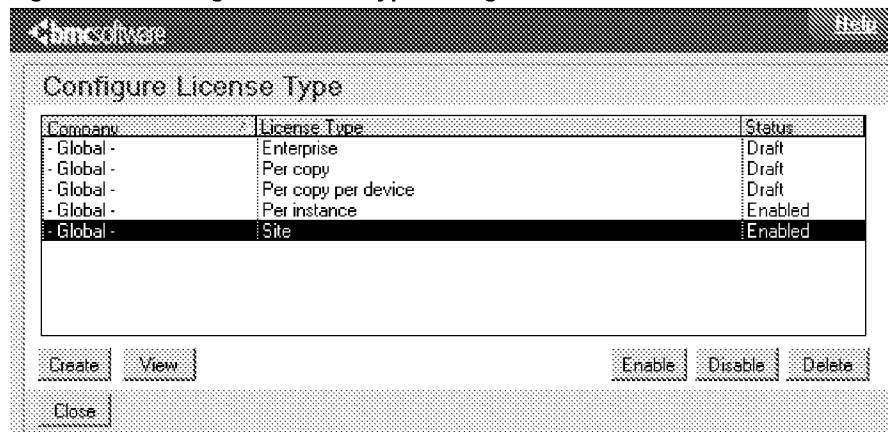
This section provides an example to show you how to create a custom license type in basic mode based on the default Site license type. For more information about default license types, see “Default system license types” on page 418.

► **To create the site example license type in basic mode**

- 1 From the Application Administration Console, click the Custom Configuration tab.
- 2 From the Application Settings list, choose Asset Management > Advanced Options > Configure License Type, and then click Open.

The Configure License Type dialog box appears as shown in Figure B-1.

Figure B-1: Configure License Type dialog box



- 3 Click Create.
 - The license type wizard starts.
- 4 Update the Create License Type page as follows.
 - a Select Global as the company. This makes the license type available to all companies.
 - b Enter a name for the license type. For example, you would enter Site Example.

NOTE

Ensure that you do not name this license type Site, since that license type already exists in the system.

- c Enter a description for the license type. For example, you would enter Example site license that includes all software from a manufacturer being used by a specific site.
 - By default, the status is set to Draft.
- d For Groupable, select No.

- e For License Type Mode, select Basic.

NOTE

After a license type is created, it has to be enabled. This example shows you how in a later step.

Figure B-2 shows what the Create License Type page should look like.

Figure B-2: Create License Type page filled in for the example

The screenshot displays the 'Create License Type' page in BMC REMEDY IT SERVICE MANAGEMENT - Asset Management. The page title is 'Create License Type'. On the left, there is a navigation menu with the following items: 'Create License' (selected), 'Connection Questions', 'Connection Mapping', 'Compliance Questions', and 'Compliance Actions'. The main form area is titled 'License Type Information' and contains the following fields:

- Company:** Global
- License Type:** Site Example (with a 'Manage Locales' button next to it)
- Description:** Example site license that includes all software from a manufacturer being used by a specific site
- Status:** Draft
- Groupable:** No
- License Type Mode:** Basic

At the bottom of the page, there are three buttons: 'Close' on the left, 'Previous' in the center, and 'Next >>' on the right.

- 5 Click Next.
- 6 Update the Connection Questions page as follows:
 - a For Question, enter text that prompts for the product name. For example, you would enter Enter certificate site.
 - b For the field type, select Char.
 - c For the CI Grouping, select Computer System.
Computer System is selected instead of Software because computer systems, and not CIs, are usually updated with Site, since there are typically fewer of them.
 - d Click the Sequence up arrow to make the sequence 1.

Figure B-3 shows what the Connection Questions page should look like with the information filled in.

Figure B-3: Connection question added in the example



a Click Add.

The connection question is added to the list.

7 Click Next.

8 Update the Map Connection Attributes page of the license type wizard as follows:

a For the CI Type, select Computer System.

b For the Attribute, you would select the database name. For example, you would select Site.

Since the question requests the site name be entered, this mapping needs to be made so the system can return the list of computer systems that have a site name that matches the site name entered (in answer to the connection question created earlier) when the certificate is created. The next sections in this example show you how to build the compliance questions and actions that are used to determine compliance.

c Click Map Attribute.

The mapping is added as shown in Figure B-4.

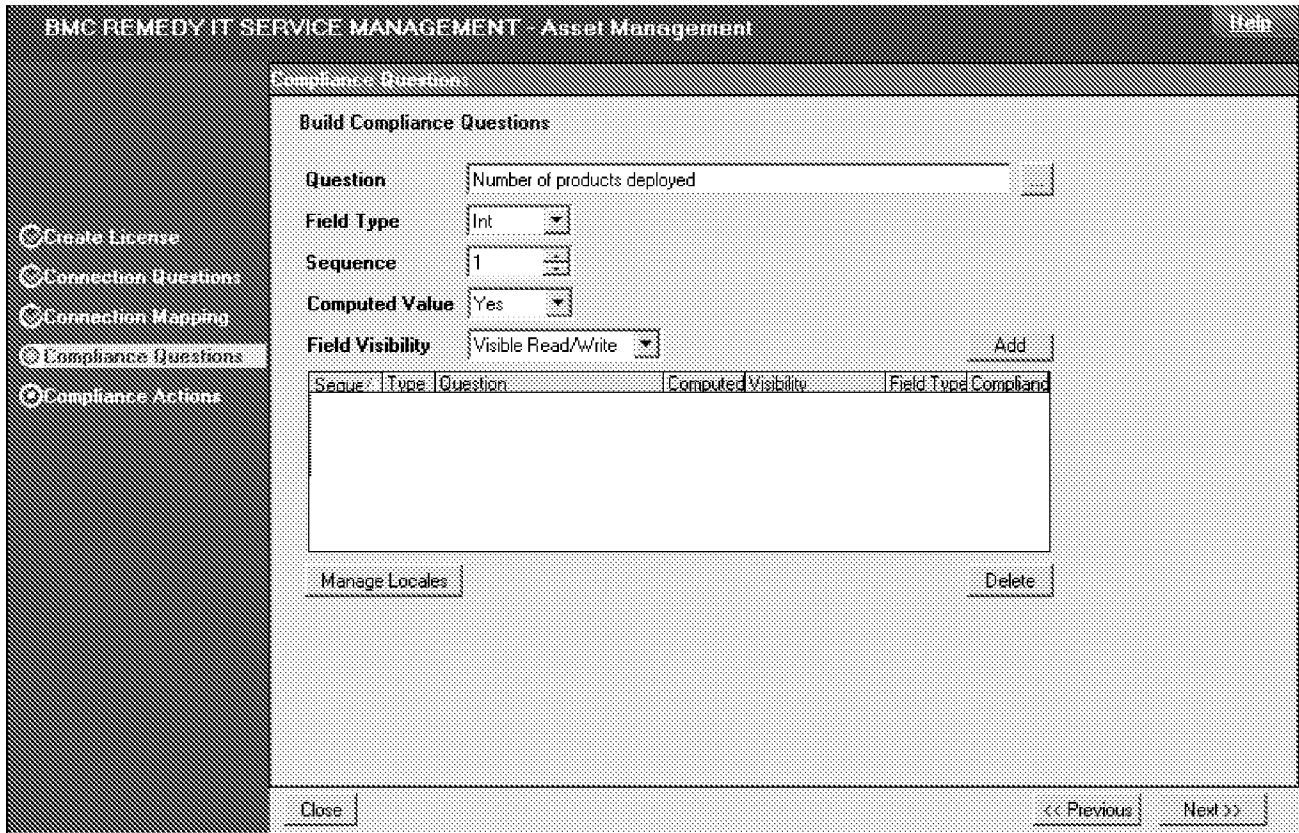
Figure B-4: Map Connection Attributes page with CI Type and Attribute specified

The screenshot shows the 'Map Connection Attributes' page in BMC Remedy IT Service Management. The main area is titled 'Build Connection Mapping'. It features a table with three columns: 'Type', 'Question', and 'CI Grouping'. The first row contains the values 'Char', 'Enter the site for w', and 'Computer System'. To the right of this table, there are two dropdown menus: 'CI Type' (set to 'Computer System') and 'Attribute' (set to 'Site'). Below these is a 'Map Attribute' button. Underneath is a larger table with five columns: 'Question', 'Source', 'CI Grouping', 'Class', and 'Attribute'. This table is currently empty. At the bottom right of this table is a 'Delete' button. The page has a left sidebar with navigation options: 'Create License', 'Connection Questions', 'Connection Mapping' (highlighted), 'Compliance Questions', and 'Compliance Actions'. At the bottom of the page are 'Close', '<< Previous', and 'Next >>' buttons.

- 9 Click Next.
- 10 On the Compliance Questions page that appears, build the first compliance question as follows:
 - a Enter the question. For this example, you would enter Number of products deployed.
 - b For Field Type, select Int.
 - c For Sequence, select 1.
This indicates the sequence in which the question is asked when the license certificate is created.
 - d Since this compliance question is updated later by an update action, select Yes for Computed Value.
 - e For the Field Visibility, select Visible Read/Write.
Since this field is being update by a value entered when the license certificate is created, the field visibility needs to be set to read / write access.

Figure B-5 shows how the Compliance Questions page should look like.

Figure B-5: Build Compliance Questions page filled out



f Click Add.

The question is added to the list.

11 Click Next.

In the next section of the example, you will be shown how to create the actions that work with the compliance question to determine whether or not the list of certificates retrieved by the connection question and its mapping are compliant.

12 On the Compliance Actions page of the create license wizard, select Get in the Action field and then click Create.

The Get Action dialog box is displayed.

13 Build the Get action:

a Enter the name of the Get action. For this example, you would use Get deployed products, because this action is retrieving the number of deployed licenses from the list of certificates returned by the connection question and mapping configured earlier.

b Select AR as the source from which to retrieve the information.

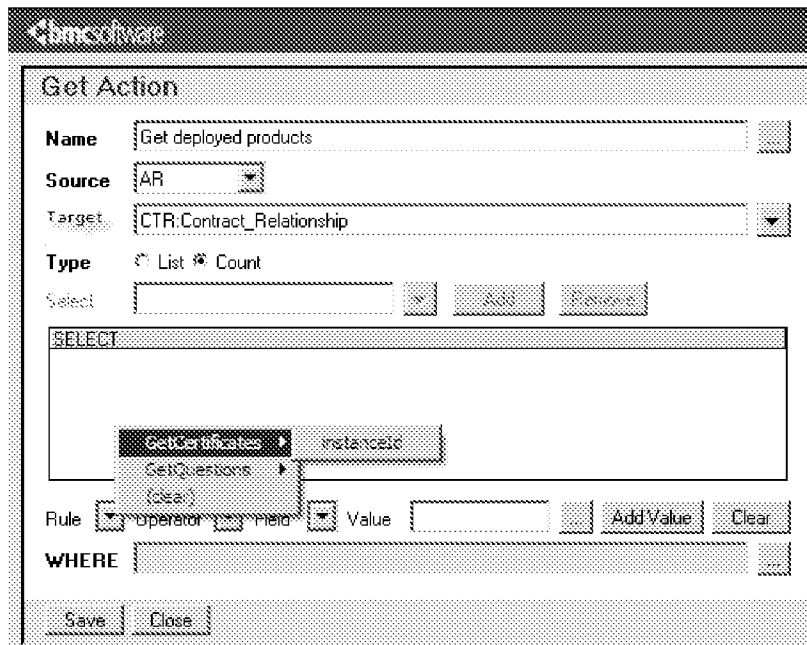
- c In the Target field, enter the form name from which to retrieve the information. In this example we are need to retrieve the information from CTR:Contract_Relationship, which holds all connection information stored for processed connection questions.

TIP

You configure a license type with the conditions that will be present at the time the license certificate is created. The form CTR:Contract_Relationships will be populated with the information that this Get action needs at that time.

- d For this license type we want to count the number of deployed licenses so select Count as the type.
- e From the Rule list, select GetCertificates > instanceId, as shown in Figure B-6, to start the rule count to retrieve the number of license certificates.

Figure B-6: Get Action dialog box filled in and with WHERE qualifier being built



- f From the Operator list, select =.
- g From the Value list, select Child Instance ID*.
- h The WHERE field now contains GetCertificates.instanceId = 'Child_Instance_ID' as shown in Figure B-7.

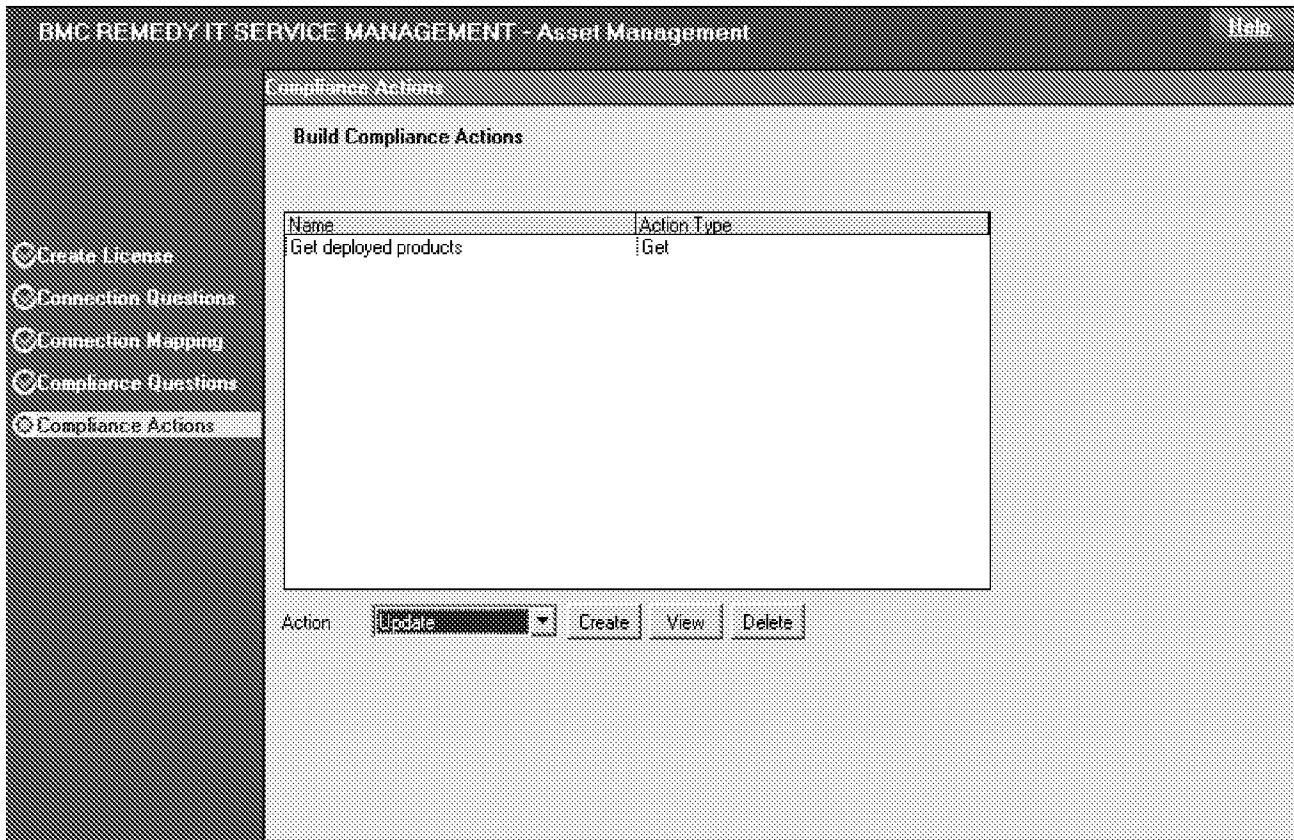
Figure B-7: WHERE field updated with built WHERE qualifier



- i Click Save.

The Compliance Actions page of the license wizard is redisplayed and the Get action appears in the list as shown in Figure B-8.

Figure B-8: Compliance Actions page showing the Get action created



In order to display the results of the Get action and set the compliance flag, you must build an update action.

- 14 From the Action list, select Update and click Create.
- 15 Build the update action as follows:
 - a Enter the name. For this example, you would enter Update Compliance.
 - b For the Source select Compliance Question.
 - c From the Set list, select Number of products deployed. This value indicates that the question field should be updated based on the count retrieved in the Get.
 - d In the Set Value From field, select Rule.