

EMC Smarts Openness

Options for Customization

Abstract: While many management solutions use customization to develop basic functionality, EMC[®] Smarts[®] solutions are built on a powerful platform and use customization to enhance basic functionality to meet specific business requirements. This paper discusses the openness of Smarts technology and presents several dimensions along which EMC Smarts, its partners, and its customers can enhance existing Smarts solutions and develop new ones.



November 28, 2005
Originally Published December 2003

Copyright © 2005 EMC Corporation. All rights reserved.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED “AS IS.” EMC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

Sample Trademark page below.

EMC², EMC, ApplicationXtender, Celerra, CentraStar, CLARAlert, CLARiiON, Connectrix, Dantz, Direct Matrix Architecture, DiskXtender, Documentum, EmailXtender, EmailXtract, HighRoad, Legato, Navisphere, PowerPath, RepliStor, ResourcePak, Retrospect, Smarts, SnapView/IP, SRDF, Symmetrix, TimeFinder, VisualSAN, and where information lives are registered trademarks and EMC ControlCenter, EMC Developers Program, EMC OnCourse, EMC Proven, EMC Snap, EMC Storage Administrator, Access Logix, ArchiveXtender, Automated Resource Manager, AutoSwap, AVALONidm, C-Clip, Celerra Replicator, Centera, CLARevent, CopyCross, CopyPoint, DatabaseXtender, Direct Matrix, DiskXtender 2000, EDM, E-Lab, EmailXaminer, Engenuity, eRoom, FarPoint, FLARE, FullTime, InfoMover, MirrorView, NetWin, NetWorker, OnAlert, OpenScale, Powerlink, RepliCare, SafeLine, SAN Advisor, SAN Copy, SAN Manager, SDMS, Smarts, SnapSure, SnapView, StorageScope, SupportMate, SymmAPI, SymmEnabler, Symmetrix DMX, and VisualSRM are trademarks of EMC Corporation. All other trademarks used herein are the property of their respective owners.

All other brand names are trademarks or registered trademarks of their respective owners.

Part Number

Table of Contents

Introduction	4
Defining Management Policies	5
Discovery	5
Defining Groups and Policies for the Network	5
Defining Groups and Policies for Applications	7
Defining Business Objects with the Topology Browser	10
Defining Automated Responses, Workflow, and Escalation Policies	13
Adding Support for New Vendor Products by Extending the Mediation Layer.....	14
Integrating Smarts with Other Software.....	16
Importing and Exporting Events, Topology, and Results of Analysis.....	16
Leveraging Information in Smarts to Build New Management Applications	18
Extending ICIM Library Models to New Domains	20
Conclusion.....	21
References.....	21

Introduction

Most management systems rely on expensive, never-ending rules development to bolt analysis onto rudimentary event monitoring and filtering platforms. These systems require custom rules to accomplish even the most basic event management tasks, and these rules must be updated constantly as the infrastructure changes.

In contrast, EMC Smarts' unique platform architecture enables powerful analytics to be built into commercial off-the-shelf solutions. These "out-of-the-box analytics" adjust automatically to the managed system at start-up and adapt dynamically as the environment changes. As a result, EMC Smarts solutions deliver immediate time-to-value, assuring the highest quality of service, and the fastest ROI in the industry. The analytics embedded in EMC Smarts solutions are far more powerful and cost-effective than rules-based "add-on" intelligence, because no matter how high the investment, rules that must be modified with every system change can never stay current.

With such unprecedented intelligence and functionality built-in, one might expect that EMC Smarts solutions are fixed and do not lend themselves to customization or enhancement. In fact, exactly the opposite is true. The key difference is that with Smarts, customization is used not to develop basic functionality, but rather to enhance this functionality by leveraging the knowledge and capabilities resident in Smarts to respond to unique business needs.

This paper presents several dimensions along which Smarts, its partners, and customers can enhance existing EMC Smarts solutions and develop new ones. The enhancements covered in this paper include:

1. Customize/define management policies for monitoring, analysis, and problem management.
2. Define automated responses and escalation policies and associate them with instances of events and service-affecting *authentic problems*TM.
3. Add support for new vendor products to the Smarts Mediation Layer.
4. Integrate EMC Smarts solutions with third-party systems by defining adapters to import and export information including topology, data, and events.
5. Leverage the information maintained in the EMC Smarts ICIM Common Information ModelTM Repository to develop new applications such as inventory and asset management, provisioning, capacity planning, and others.
6. Refine existing ICIM library models to enrich the automated analytics, or create new classes of managed entities to extend Smarts analysis to new technology domains.

Defining Management Policies

The EMC Smarts Console provides an easy, user-friendly way for users to define management policies and make them effective immediately—without disrupting service. No programming experience is required. While other management systems offer only “fixed” policies with no ability to fine-tune or set parameters by group, Smarts provides exceptional openness. With a few simple console interactions, users can group managed entities and define policies on a per-group basis. Groups can be set up based on any property or discovered attribute.

Discovery

Defining management policies starts with discovery. While auto-discovery must provide fine control of what is discovered, it is equally important to control what is not discovered. Smarts provides console-based controls for setting auto-discovery parameters in fine detail. Auto-discovery can also be manually initiated, periodically scheduled, or automatically triggered on receipt of network change events.

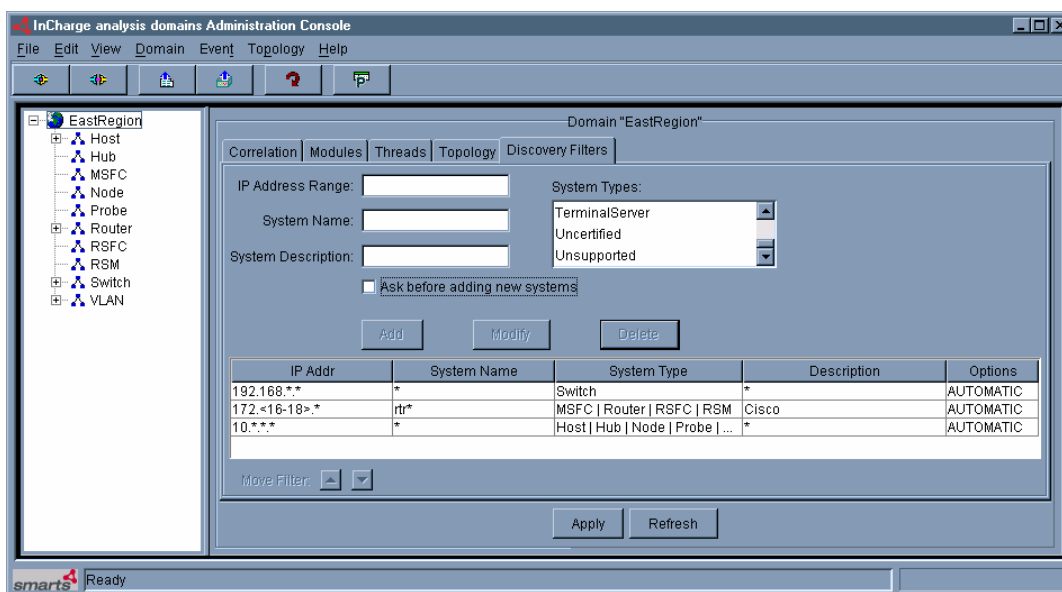


Figure 1: Setting Discovery Parameters

Defining Groups and Policies for the Network

Once the network has been discovered, users can build a highly customized polling policy to meet specific infrastructure requirements. For example, in Figure 2, a new group called “Core 1 Gb” has been created by copying the “1 Gb Ethernet” group and modifying the membership criteria so that only interfaces from a specific network device can be in the group. Because groups are defined by matching criteria, new objects are automatically added to their corresponding groups.

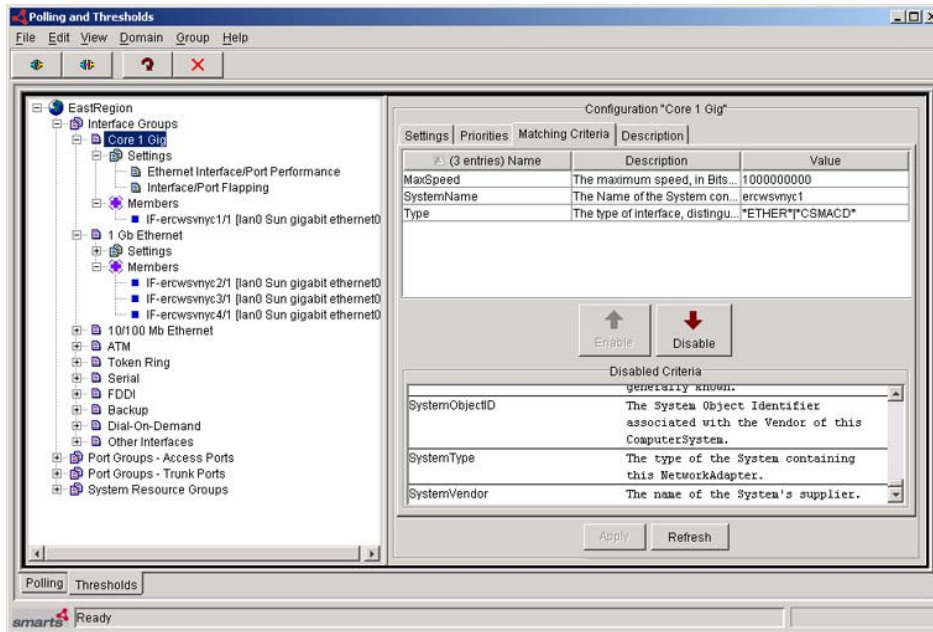


Figure 2: Creation of Group “Core 1 Gb” with Specific Membership Criteria

Smarts can then be configured for a unique policy for the interface(s) that belongs to this group. Examples of policies include the types of issues to monitor, what data to poll, how often to poll, etc. Figure 3 illustrates how to specify that for the group called Routers, Smarts is to perform connectivity polling (pinging IP interfaces and polling SNMP status), environment polling (polling temperature and voltage), port and interface performance (polling interface utilization/errors), and CPU and memory polling (CPU, memory, and buffer utilization/errors).

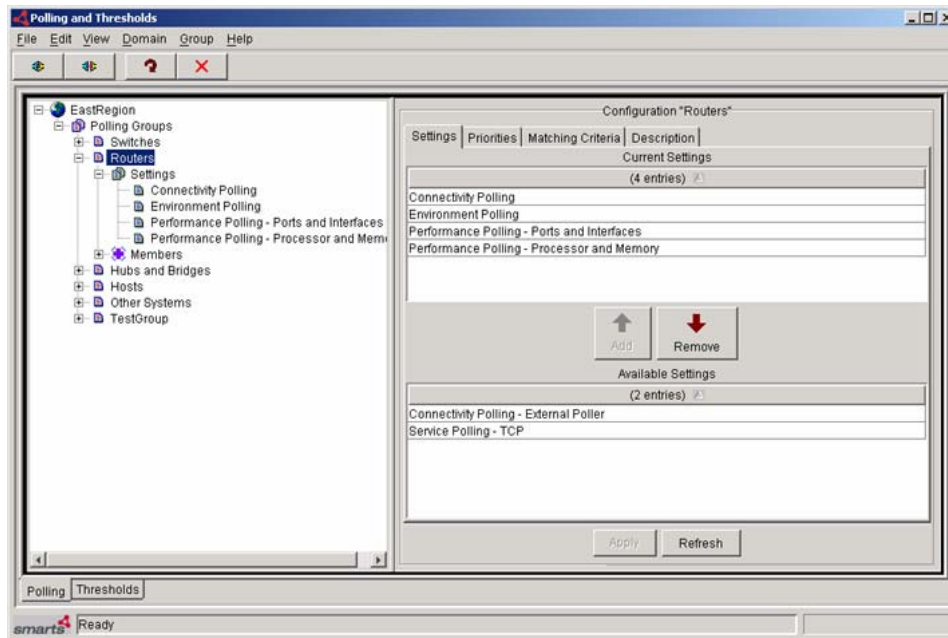


Figure 3: Defining Problems to Monitor for Router Group

Users can set the details for each policy for each group. For example, in Figure 4, Smarts is configured with analysis enabled, and the polling interval is set to 240 seconds.

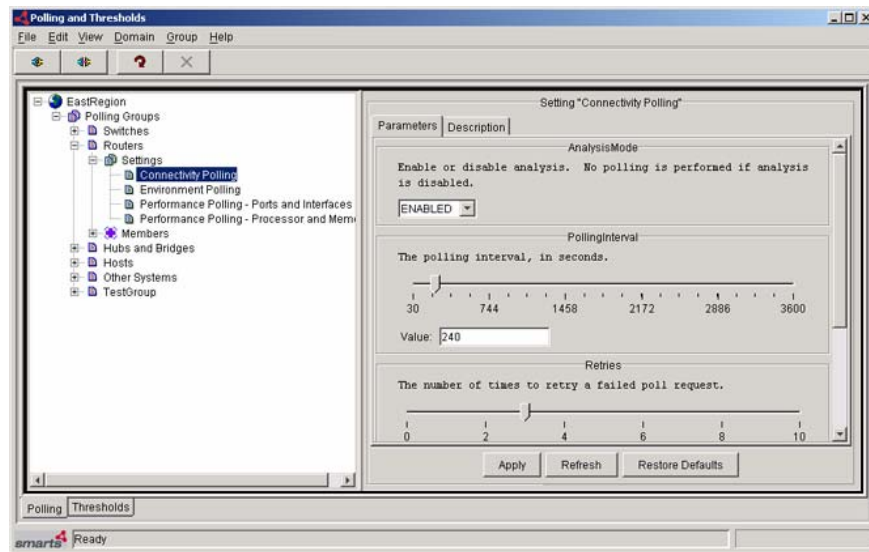


Figure 4: Turning on Analysis and Setting Polling Intervals

Defining Groups and Policies for Applications

The same flexibility to define groups and set policies is available on the application side. Figure 5 shows how a Smarts Application Servers group is defined by setting membership criteria to be Application Servers that include “smarts” in their names and system names.

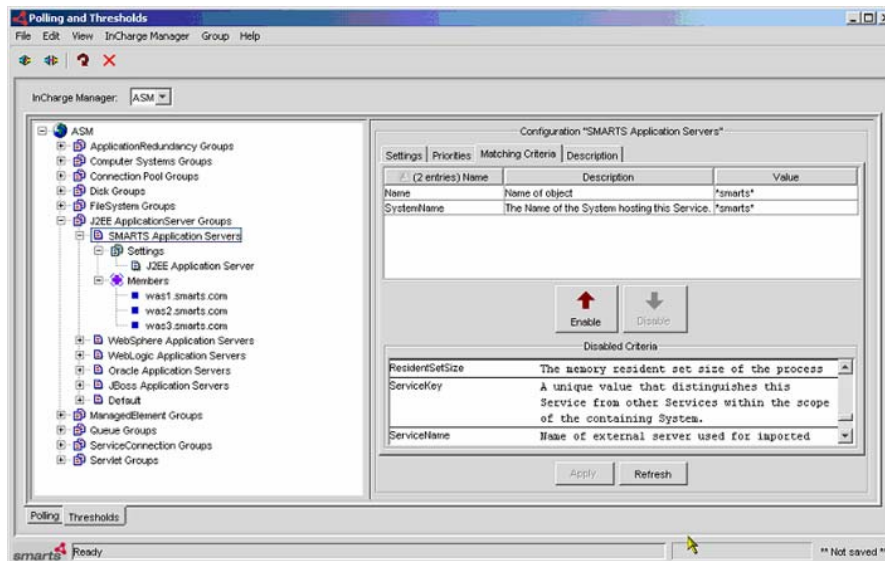


Figure 5: Defining Group Membership: Smarts Application Servers Group

Users also can define management policies for each applications group. Figure 6 illustrates how to specify that for the group called Smarts Application Servers, the system is to monitor the internal statistics of the of the Application Servers (J2EE Application Server) and the performance of the process resources (Process resources).

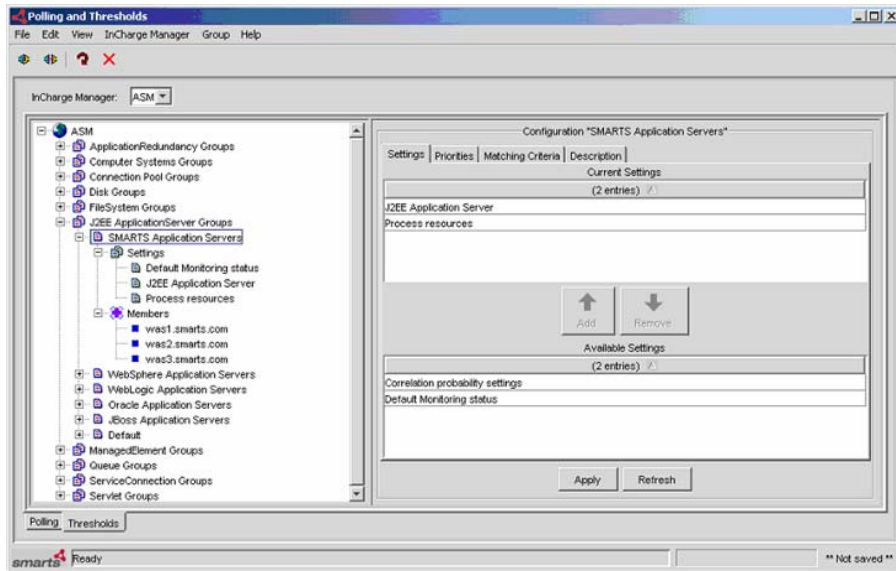


Figure 6: Defining Problems to Monitor for Application Servers Group

Details can be set for each policy for each group. In Figure 7, the Smarts system is configured with the TCP polling interval set to 240 seconds and three retries for each attempt.

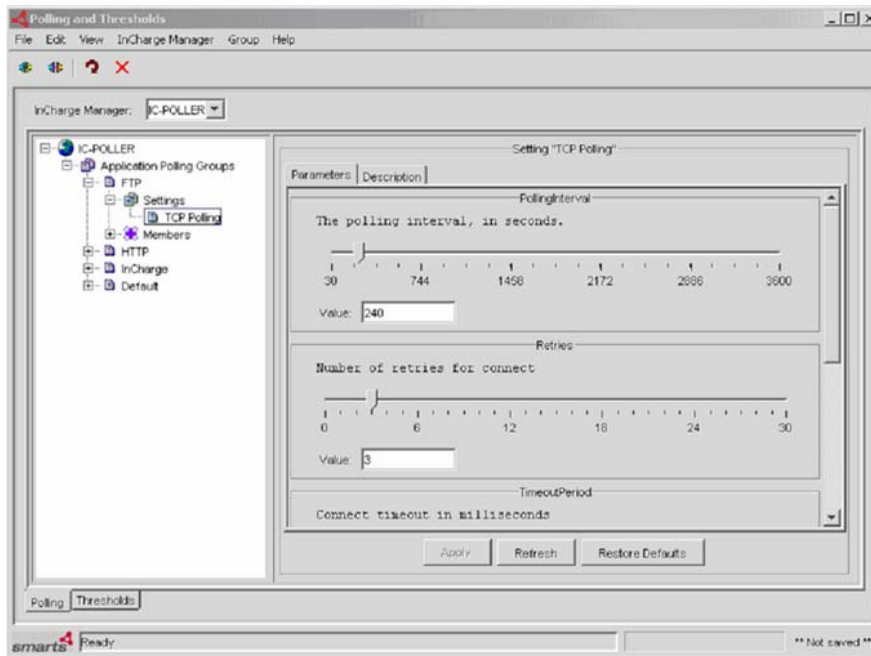


Figure 7: Setting Retries and Timeout for Polling

The EMC Smarts Global Console makes it easy for users to totally customize threshold analysis. In Figure 8, the Smarts system has been configured so that when garbage collection processing takes more than 5 percent of the JVM, a notification is raised. Similarly, if the number of active threads in the pool exceeds 100, a notification is raised.

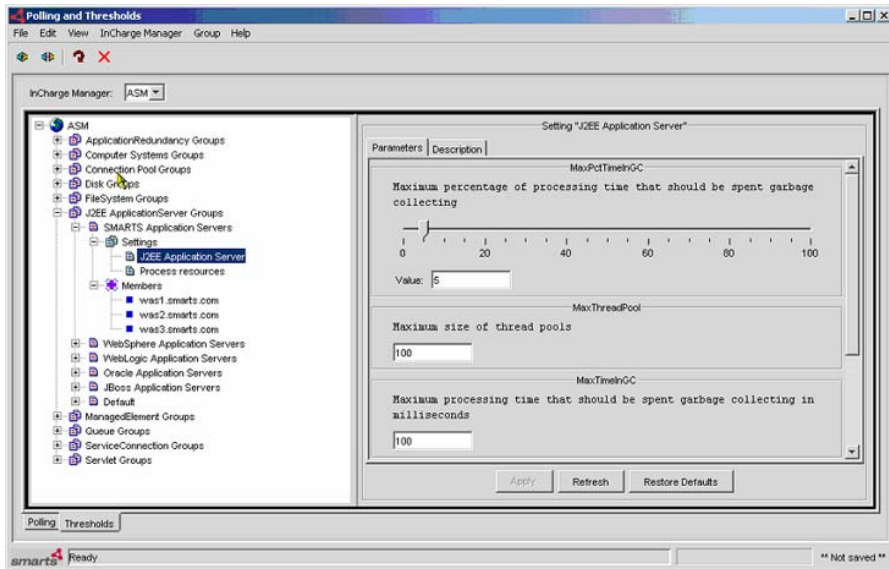


Figure 8: Setting Garbage Collection and Thread Pool Thresholds

EMC Smarts Service Assurance Manager allows users to define groups to match high-level business objects. Figure 9 shows how the Global Console can be used to define top-level groups, sub-groups, and members within each group. Figure 10 shows how Smarts displays custom geographic groupings in maps.

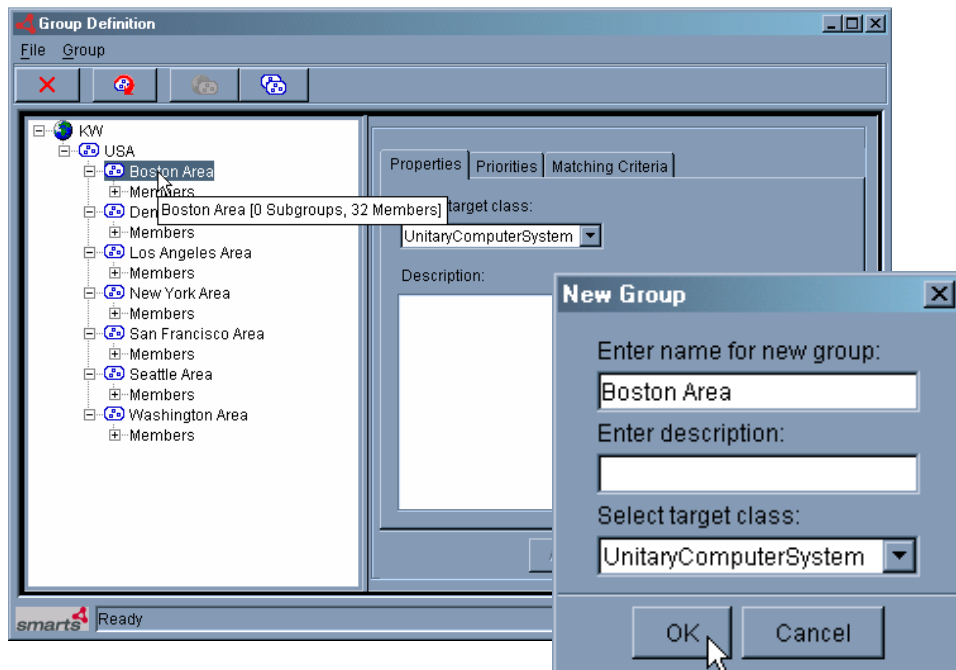


Figure 9: Defining Geographic Groups

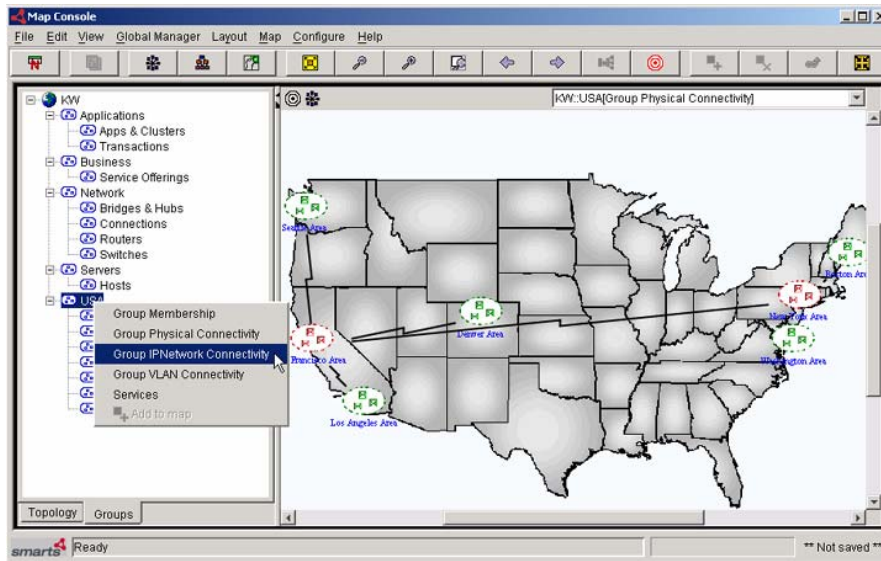


Figure 10: Geographic Groups Displayed in Custom Map

Defining Business Objects with the Topology Browser

The EMC Smarts Topology Browser is designed specifically for viewing the topology of any Smarts solution. It also provides an easy way to add new application services components to the Smarts Repository and to display them in maps.

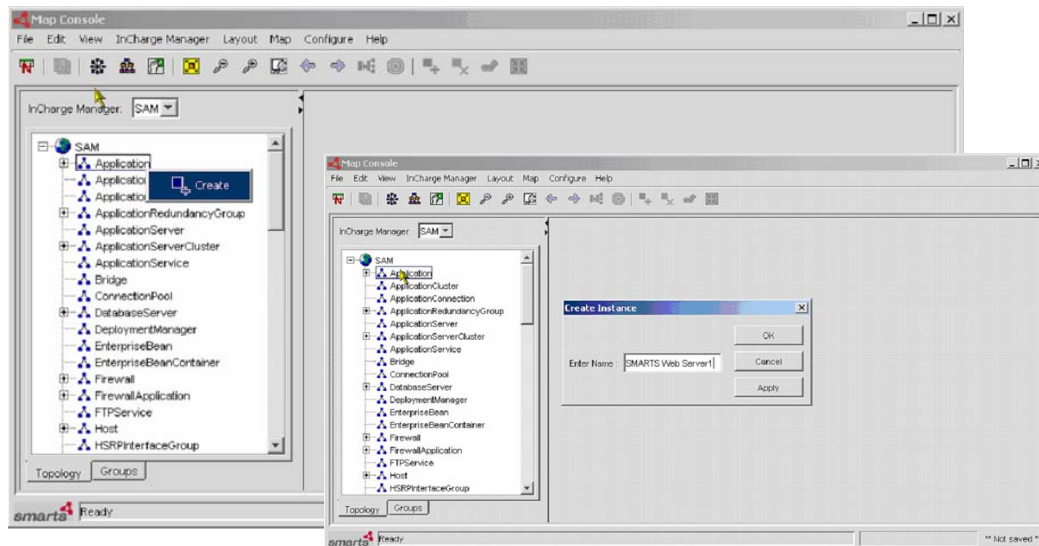


Figure 11: Creating an Application Called Smarts Web Server1

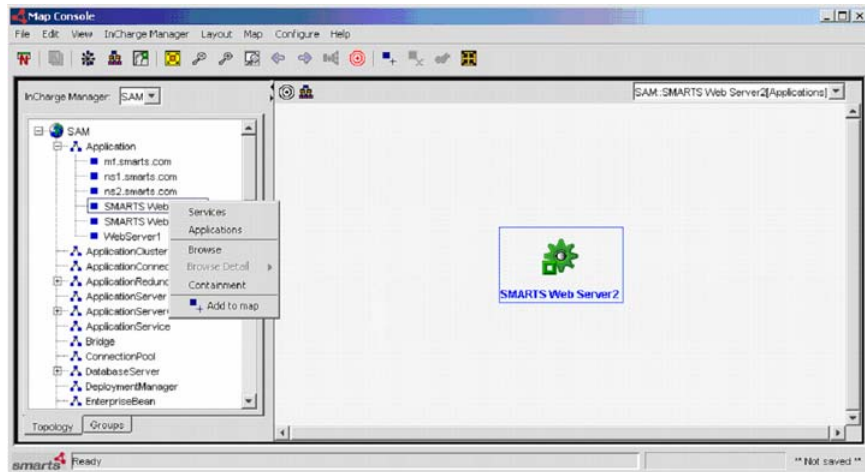


Figure 12: Adding Smarts Web Servers to the Map

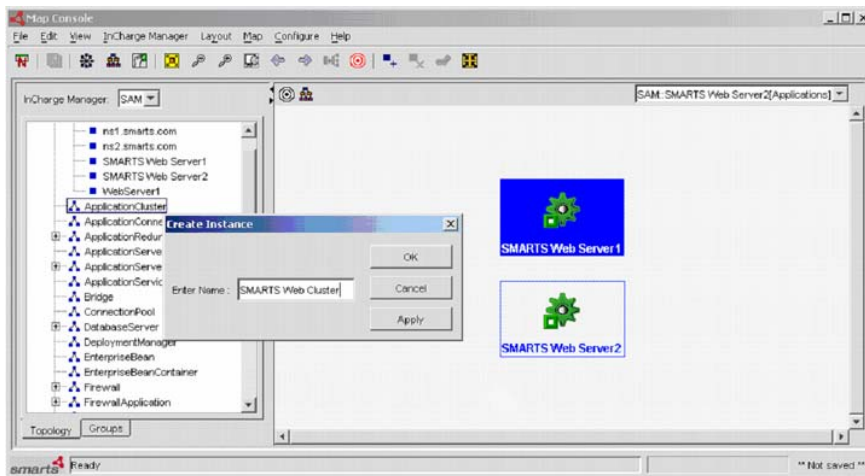


Figure 13: Creating Smarts Web Cluster

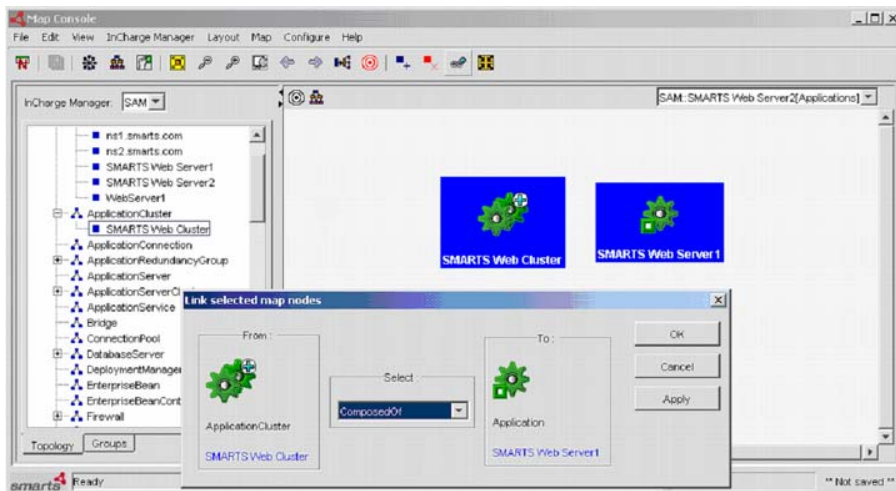


Figure 14: Associating Smarts Web Cluster to its Components

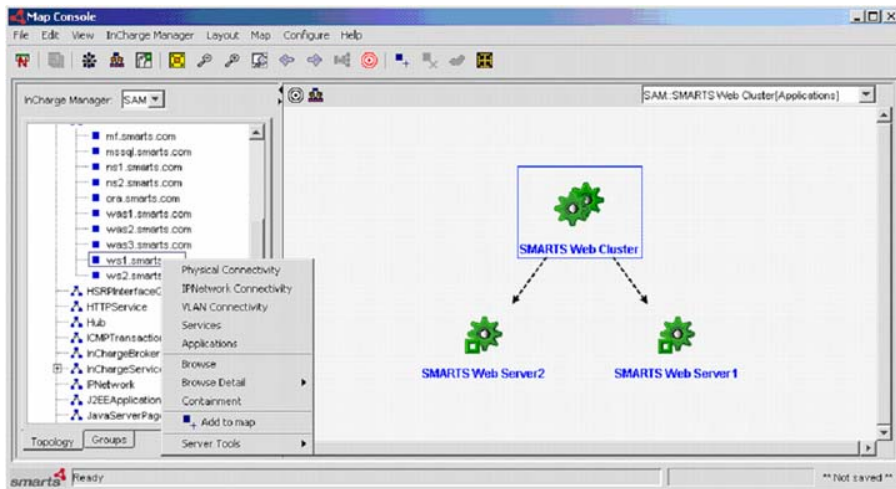
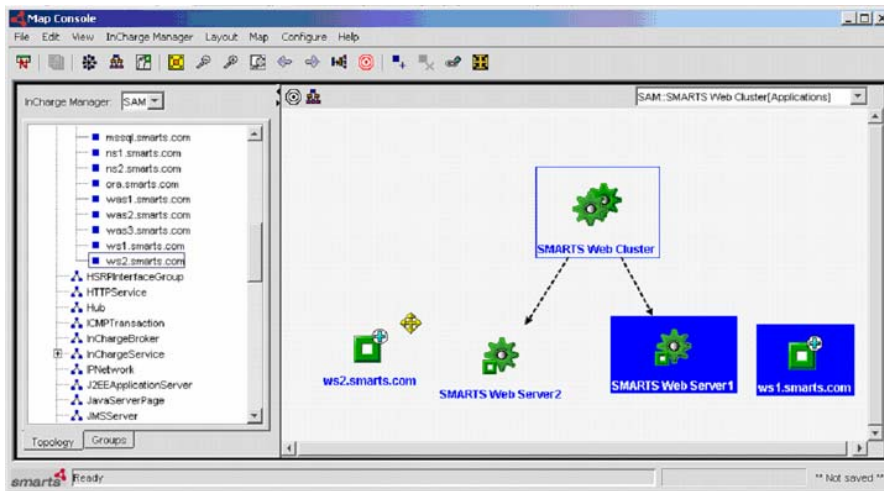
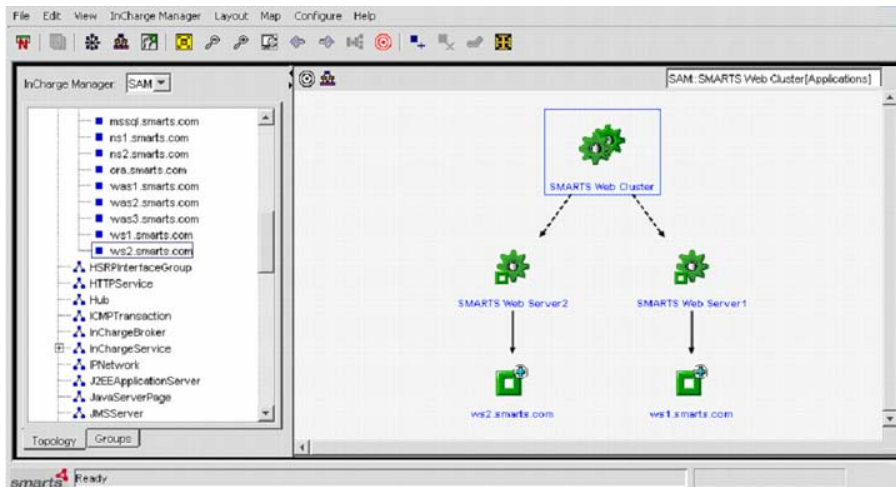


Figure 15: Adding Hosts to the Map



Figures 16-17: Associating Smarts Web Servers with Their Hosts



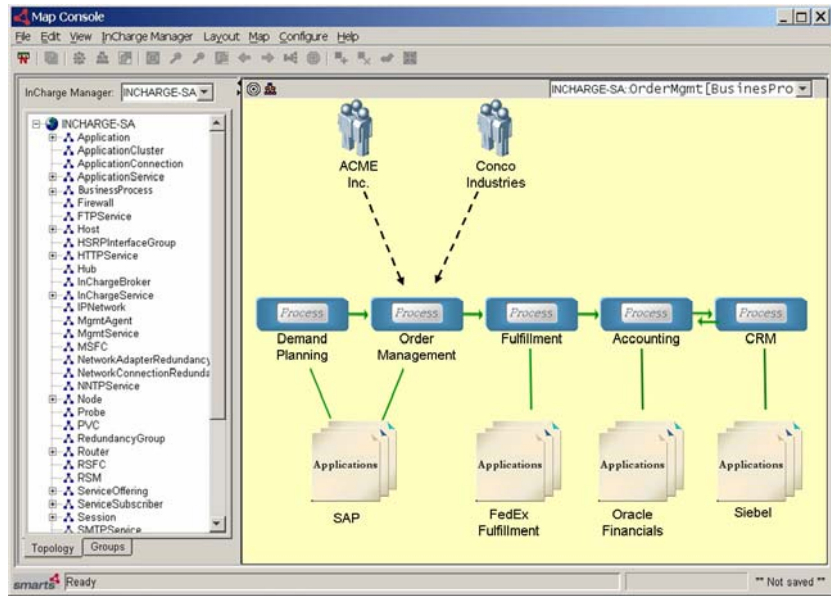


Figure 18: Business Process View

Defining Automated Responses, Workflow, and Escalation Policies

Smarts' client-server architecture makes it easy to associate automated responses with notified events and authentic problems. The client interface includes an operation to subscribe to be notified of events and problems. Upon receipt of a particular notification or category of notifications, an open-ended set of actions can be invoked. These actions can involve a variety of operations, including arbitrary queries to the Smarts Repository.

Ack	Active	Class	Name	First Notify	Count	Source	Impact	Event Text
No	Yes	Application	BankingApp:1	29 Apr 13:39:09	1	ASM_SA	1314	The application is not working.
No	Yes	Application	Payments_and_Invoicing:1	29 Apr 13:39:06	1	ASM_SA	3	The application is not working.
No	Yes	Switch	arcasmyc:1	29 Apr 13:39:28	1	EastRegion	3307	System is unavailable.
No	Yes	Card	arcasmyc:1	29 Apr 13:39:11	1	WestRegion	1310	Card has failed. Either an indication from the instrument...
No	Yes	Host	arcasmyc:1	29 Apr 13:39:25	1	EastRegion	0	Impact of NOTIFICATION Host_arcasmyc:1_Unexpected...
No	Yes	Host	arcasmyc:1	29 Apr 13:39:27	1	EastRegion	0	No Services or ServiceAccessPoints on this System are...
No	Yes	Host	arcasmyc:1	29 Apr 13:39:27	1	EastRegion	0	No Services or ServiceAccessPoints on this System are...
No	Yes	Host	arcasmyc:1	29 Apr 13:39:25	1	EastRegion	0	No Services or ServiceAccessPoints on this System are...
No	Yes	Applicat	arcasmyc:1	29 Apr 13:39:35	1	ASM	0	This Application Service is impacted by a Host problem...
No	Yes	Service	arcasmyc:1	29 Apr 13:39:26	1	EastRegion	0	No Services or ServiceAccessPoints on this System are...
No	Yes	Service	arcasmyc:1	29 Apr 13:39:15	1	EastRegion	0	No Services or ServiceAccessPoints on this System are...
No	Yes	Transac	arcasmyc:1	29 Apr 13:39:07	1	SiteScope	0	Impact of NOTIFICATION Transaction_LWR_APP_System...
No	Yes	Transac	arcasmyc:1	29 Apr 13:39:09	1	SiteScope	0	No Services or ServiceAccessPoints on this System are...
No	Yes	SQL Ser	arcasmyc:1	29 Apr 13:39:35	1	ASM	0	This Application Service is impacted by a Host problem...
No	Yes	Applicat	arcasmyc:1	29 Apr 13:39:27	1	ASM	0	This Application Service is impacted by a Host problem...
No	Yes	SQL Ser	arcasmyc:1	29 Apr 13:39:35	1	ASM	0	This Application Service is impacted by a Host problem...
No	Yes	Host	arcasmyc:1	29 Apr 13:39:27	1	EastRegion	0	No Services or ServiceAccessPoints on this System are...
No	Yes	Host	arcasmyc:1	29 Apr 13:39:27	1	EastRegion	0	No Services or ServiceAccessPoints on this System are...
No	Yes	Switch	arcasmyc:1	29 Apr 13:39:28	1	EastRegion	0	No Services or ServiceAccessPoints on this System are...
No	Yes	Applicat	arcasmyc:1	29 Apr 13:39:26	1	ASM	0	This Application Service is impacted by a Host problem...
No	Yes	ServiceSubscri...	arcasmyc:1	29 Apr 13:39:26	1	EastRegion	0	Impact of NOTIFICATION Host_arcasmyc:1_Unexpected...
No	Yes	Host	arcasmyc:1	29 Apr 13:39:25	1	EastRegion	0	No Services or ServiceAccessPoints on this System are...
No	Yes	ServiceSubscri...	arcasmyc:1	29 Apr 13:39:26	1	EastRegion	0	Impact of NOTIFICATION Host_arcasmyc:1_Unexpected...
No	Yes	HTTPService	arcasmyc:1	29 Apr 13:39:27	1	ASM	0	This Application Service is impacted by a Host problem...
No	Yes	SQL Service	arcasmyc:1	29 Apr 13:39:27	1	ASM	0	This Application Service is impacted by a Host problem...
No	Yes	ApplicationClust...	arcasmyc:1	29 Apr 13:39:28	1	ASM	0	This application cluster is at risk of meeting service avail...

Figure 19: Acting on an Alarm in the Notification Log

Workflow automation is a critical component of automated operations; it allows organizations to define and enforce the processes that follow the identification of an incident that requires action. Smarts allows defining workflow policies for each user-defined group. Policies can specify for different groups of notifications, levels of escalation, and corresponding actions at each level. Smarts users can define open-ended escalation policies using any

information in the Smarts Repository. For example, escalations can be time-based, count-based, or based on any attribute accessible via the Repository. Escalations can be defined via the EMC Smarts Global Console.

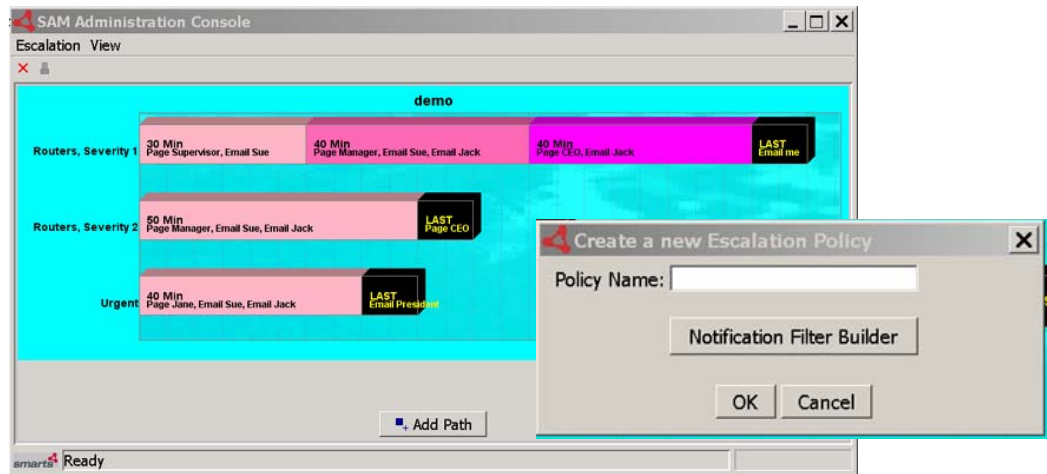


Figure 20: Using the *InCharge* Global Console to Define Escalation Policies

Adding Support for New Vendor Products by Extending the Mediation Layer

Smarts mediation technology makes it possible for the Smarts system to always stay current as new products enter the market. Partners and customers can easily extend Smarts' coverage without developing any code.

The EMC Smarts ICIM Common Information Model provides a common semantic framework for all Smarts solutions. Each Smarts application comes with a library of generic ICIM models of logical and physical objects for the managed domain. The ICIM models are abstract; the mediation layer's purpose is to map these generic models to real-world products.

Smarts automatically discovers and manages any device that implements MIB-II to a certain level of detail. However, to enhance the level of Smarts support to cover more detailed discovery and analysis, Smarts employs certifications.

Smarts' mediation layer is dynamic and modular, with product-specific information organized in certifications, one per vendor product. A certification maps the attributes and relationships in the abstract ICIM model to their concrete counterparts in the vendor product. Beyond MIB-II, certifications involve the use of other standard MIBs and enterprise-specific MIBs that contain the detailed information Smarts needs.

Adding support for a new vendor product amounts to developing a new certification and inserting it into the mediation layer. When the managed entity is an SNMP device whose MIB is properly documented, this is a simple, straightforward process: it requires a scan of the device's MIB to generate a file to inform Smarts where in the MIB it can find the information it needs to populate its generic object model. Certifications are simple ASCII files, typically made via cut-and-paste of other mediation layer entries with some modifications. See Figure 21 for a sample template.

A certification extends Smarts to:

- Automatically discover in detail all of the objects and relationships pertinent to the added device.
- Automatically monitor the device in real time for data and events that matter.
- Automatically identify exceptional behaviors and *authentic problems*.

EMC Smarts publishes a new incremental device support (IDS) release at least four times each year. Each IDS adds 30-60 new certified devices to the mediation layer. EMC Smarts customers receive IDS updates as part of their software maintenance, and may also request certification of specific devices. Alternatively, users may complete the

certification process in the field (see Figure 21). The IDS can be installed directly over a current Smarts application—no reinstallation is required.

```
# oid2type_Field.conf.template
#
# Each entry starts with the sysObjectID of the device followed by "{".
# Each entry ends with "}"
#
# TYPE:
# The type of entity. This field is mandatory.
# It must appear right after the sysObjectID.
#
# VENDOR:
# The manufacturer of this device, e.g. IBM, Cisco, Microsoft, # etc.
# This field is mandatory. It must appear right after the TYPE field.
# The value of VENDOR must be a single word.
#
# MODEL:
# The vendor model designation, e.g. PowerPC, etc.
# This field is optional.
#
# CERTIFICATION:
# The value should always be TEMPLATE which means that this system is
# recognized by the InCharge Domain Manager, but there is no knowledge
# about support for various MIBS used in the discovery process.
#
# INSTRUMENTATION:
# Interface-Fault = MIB2
# This specifies which MIB variables Smarts polls to
# get the interface status. MIB2 means Smarts queries ifAdminStatus
# and ifOperStatus for the interface status.
#
# Interface-Performance = MIB2
# This specifies that Smarts polls the ifInOctets, ifOutOctets, and a
# few other counters in the ifTable to get the interface utilization
# and other statistics.
#
#####
#
# This is an example of a host (TYPE = Host)
# IBM PowerPC Personal Computer
# 1.3.6.1.4.1.2.3.1.2.1.1.3 {
# TYPE = Host
# VENDOR = IBM
# MODEL = PowerPC
# CERTIFICATION = TEMPLATE
#
# INSTRUMENTATION:
# Interface-Fault = MIB2
# Interface-Performance = MIB2
# }
```

Figure 21: Sample Template for Device Certification

Integrating Smarts with Other Software

Importing and Exporting Events, Topology, and Results of Analysis

Smarts' open architecture enables topology, status data, and events—including authentic problems—to be imported and exported to and from the ICIM Repository. The number and type of sources and destinations are unlimited.

Information can be imported or exported to and from the ICIM Repository programmatically through EMC Smarts Adapters. These adapters link Smarts to management tools and OSS/BSS systems that exist in a managed environment management solution. EMC Smarts Adapters deliver events, topology, inventory, and analysis results into and out of EMC Smarts Service Assurance Manager.

In addition, EMC Smarts Adapters support integrations to third-party products. Integrations include creation of a trouble tickets, sending a page/e-mail, and linkage to or launching of third-party products from a console via “right-click” integration. EMC Smarts Adapters support centralized visibility into the managed infrastructure, often integrating disparate tools into a unified infrastructure management solution.

A broad range of off-the-shelf adapters is available to integrate Smarts with element management, network management, service management, trouble-ticketing, performance reporting, application management, provisioning management, and security management systems. Customers, partners, and developers can also write their own adapters to meet specific business or technology requirements.

Adapters can leverage a variety of programming interfaces including C, C++, Java, and Perl application programming interfaces (APIs), command line interfaces, Adapter Scripting Language (ASL) scripts, and XML.

Information can also be input into the Repository manually via the EMC Smarts Global Console. The Global Console supports a graphical user interface to three basic adapters: a topology adapter to create and delete objects (such as `DatabaseServer`) or relationships (such as `HostedBy`), a data/event adapter to update the state of a Repository object, and a subscription adapter to request notification of particular events.

The **Adapter Scripting Language (ASL)** is a high-level, pattern-matching language designed to simplify development of more complex interactions with Smarts. ASL has built-in interfaces to query and update the ICIM Repository. Users can develop ASL scripts for many purposes, including:

- **Import/export information to/from the ICIM Repository.** ASL scripts can be written to interface Smarts to third-party products including element and network agents and managers, provisioning systems, automated alert systems, trouble-ticketing systems, asset and inventory managers, proprietary databases, and other products.
- **Subscribe to analysis results.** Adapters can be used to configure the events Smarts proactively looks for, and notifies of—i.e., define subscription profiles for particular operators or consoles.
- **Associate automated responses with event notifications.** ASL provides an easy way to build automated responses to notified events.

By providing high-level pattern-matching constructs in ASL, the complexity of developing interactions with Smarts is simplified dramatically. The ASL development effort is directly related to the breadth and depth of the imported information, the amount of processing in the adapters, and the openness of the third-party system. Smarts users require minimum scripting language skills to become proficient in ASL.

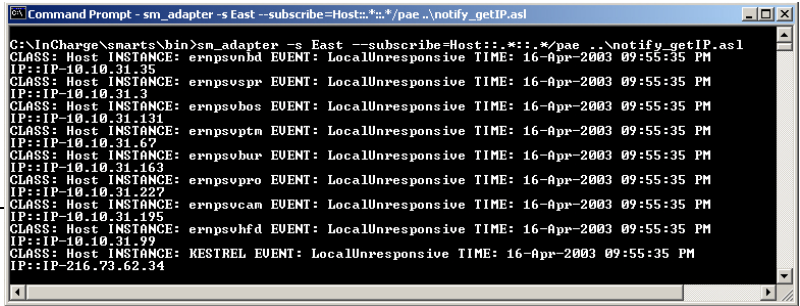
notify_getIP.asl

```
START{
  TIMESTAMP
  EVENT
}
TIMESTAMP {
timestamp: integer fs
} do {
timestring = time(timestamp);
}
EVENT{
  "NOTIFY"      fs NOTIFY
}
NOTIFY {
class: word      fs
instance: word   fs
event: word      fs
certainty: word  fs eol
} do {
print("CLASS: ".class." INSTANCE: ".instance." EVENT: ".event." TIME: ".timestring);

deviceObj = object(instance);

ipList = deviceObj->getIPs();

foreach ip (ipList) {
  print(ip);
}
```



```
C:\InCharge\smarts\bin>sm_adapter -s East --subscribe=Host:::*/*/*pae ..\notify_getIP.asl
CLASS: Host INSTANCE: ernpsunhd EVENT: LocalUnresponsive TIME: 16-Apr-2003 09:55:35 PM
IP: IP-10.10.31.35
CLASS: Host INSTANCE: ernpsvspr EVENT: LocalUnresponsive TIME: 16-Apr-2003 09:55:35 PM
IP: IP-10.10.31.3
CLASS: Host INSTANCE: ernpsvhes EVENT: LocalUnresponsive TIME: 16-Apr-2003 09:55:35 PM
IP: IP-10.10.31.131
CLASS: Host INSTANCE: ernpsvptn EVENT: LocalUnresponsive TIME: 16-Apr-2003 09:55:35 PM
IP: IP-10.10.31.67
CLASS: Host INSTANCE: ernpsvbur EVENT: LocalUnresponsive TIME: 16-Apr-2003 09:55:35 PM
IP: IP-10.10.31.163
CLASS: Host INSTANCE: ernpsvpro EVENT: LocalUnresponsive TIME: 16-Apr-2003 09:55:35 PM
IP: IP-10.10.31.227
CLASS: Host INSTANCE: ernpsvcan EVENT: LocalUnresponsive TIME: 16-Apr-2003 09:55:35 PM
IP: IP-10.10.31.195
CLASS: Host INSTANCE: ernpsvhfd EVENT: LocalUnresponsive TIME: 16-Apr-2003 09:55:35 PM
IP: IP-10.10.31.99
CLASS: Host INSTANCE: KESTREL EVENT: LocalUnresponsive TIME: 16-Apr-2003 09:55:35 PM
IP: IP-216.73.62.34
```

Convert the timestamp from the seconds since 01/01/70 to a readable format

Parse the class, instance, event, and certainty from the notification

Get the IP addresses associated with the device instance

Iterate over the IP addresses and print each one

Figure 22: ASL Adapters Retrieve Further Information from the Topology When a Notification is Received

The **EMC Smarts Perl API** allows Perl scripts to interact with Smarts servers as “clients” and exchange information with them, manipulate the data they hold, or drive their actions. The EMC Smarts Perl API provides access to all the features of Smarts analysis modules using a syntax and logic that mirrors the EMC Smarts Adapter Scripting Language (ASL) and the dmctl command line utility, but that is familiar to Perl developers.

Perl is considered by many to be the ideal integration “glue” for both large-scale integration projects and small scripts. The language is quick and easy to learn, re-usable code is plentiful, scripts are platform-neutral, and performance typically surpasses other scripting languages.

Developers can use the EMC Smarts Perl API to write scripts that integrate Smarts with databases such as Oracle and Berkley DB, create GUI applications that work on both UNIX and Windows platforms, access MIBs using the SNMP protocol, create or interact with Web services and servers, use middleware such as Tibco Rendezvous and CORBA to access third-party applications, and much more.

Class Interface - Export / Report - Results

- Dashboard TOP | SMARTS)
- Broker: localhost:9005
- AM Domain: EastRegion
- Class: Interface

HTML Table - View as : { Excel | CSV | Perl }

AdminStatus	ConnectedVia	DisplayName	InterfaceNumber	Peer	Type	getIPs ()	getMAC ()
UP	Cable::LINK-IF-eraartbwi1 <->PORT-eraaswbwi1.1	IF-eraartbwi1 [FastEthernet0]	1	Port:PORT-eraaswbwi1.1	ETHERNETCSMACD	IP:IP-10.10.65.33	MAC:MAC-00-5B-E0-00-FA-00
UP		IF-eraartbwi2 [Serial0]	2		PROPOINTTOPOINTSERIAL		
UP	NetworkConnection::LINK-IF-eraartbwi3 <->IF-eraertiad113	IF-eraartbwi3 [Serial0.101]	3	Interface:IF-eraertiad113	FRAMERELAY	IP:IP-10.10.9.131	
UP	Cable::LINK-IF-eraartphi1 <->PORT-eraaswphi1.1	IF-eraartphi1 [FastEthernet0]	1	Port:PORT-eraaswphi1.1	ETHERNETCSMACD	IP:IP-10.10.65.129	MAC:MAC-00-5B-E0-00-EB-00
UP		IF-eraartphi2 [Serial0]	2		PROPOINTTOPOINTSERIAL		
UP	NetworkConnection::LINK-IF-eraartphi3 <->IF-eraertiad110	IF-eraartphi3 [Serial0.101]	3	Interface:IF-eraertiad110	FRAMERELAY	IP:IP-10.10.9.2	
UP	Cable::LINK-IF-eraartpit1 <->PORT-eraaswpi1.1	IF-eraartpit1 [FastEthernet0]	1	Port:PORT-eraaswpi1.1	ETHERNETCSMACD	IP:IP-10.10.65.193	MAC:MAC-00-5B-E0-00-F1-00
UP		IF-eraartpit2 [Serial0]	2		PROPOINTTOPOINTSERIAL		
UP	NetworkConnection::LINK-IF-eraartpit3 <->IF-eraertiad11	IF-eraartpit3 [Serial0.101]	3	Interface:IF-eraertiad11	FRAMERELAY	IP:IP-10.10.9.34	
UP		IF-eraaswbwi1 [sc0]	0		GENERIC	IP:IP-10.10.65.34	MAC:MAC-00-5B-E0-00-FD-01
UP		IF-eraaswlad1 [sc0]	0		GENERIC	IP:IP-10.10.63.3	MAC:MAC-00-5B-E0-00-E0-01
UP		IF-eraaswphi1 [sc0]	0		GENERIC	IP:IP-10.10.65.130	MAC:MAC-00-5B-E0-00-EB-01
UP		IF-eraaswpi1 [sc0]	0		GENERIC	IP:IP-10.10.65.194	MAC:MAC-00-5B-E0-00-F4-01

Figure 23: Results of an EMC Smarts Perl API Class Interface Report Displayed in HTML Format

Leveraging Information in Smarts to Build New Management Applications

Beyond interfacing with other software products, users can leverage the powerful Smarts technology to build entirely new applications to meet their business needs. The ICIM Repository maintains a wealth of information on the managed system that can be leveraged in a variety of applications beyond realtime operations management. These applications include inventory and asset management, provisioning and service activation, simulation, capacity planning, and many others.

The ICIM Repository provides a powerful set of operations for query and update. These operations are available through a variety of Smarts interfaces including command line, ASL, Java, C, C++, XML, Perl, and others. Users can easily add or delete instances of classes or sub-classes, change the value of attributes or relationships of any instance, create new relationships between instances, and traverse existing relationships to identify related objects. For example, it is easy to programmatically identify all applications that run on a common server, the services that use these applications, and all subscribers to a particular service offering. Figures 24 and 25 show an example of a Perl script for retrieving all `ServiceSubscribers` supported by a particular host, and how these results appear in a service map.

List-subscribers.pl

```

## use Smarts Perl API
use smarts::session;

sub getParentObj($) {
    my ($obj) = @_;

    ## attempt to get all parents of object by all known relationships
    foreach $relationship ( qw[ MemberOf HostsServices Subscribers ] ) {
        my @parents = @{$obj->{$relationship}};
        if ( defined(@parents) && scalar @parents != 0 ) {
            foreach my $parent ( sort @parents ) {
                getParentObj( $session->object($parent) );
            }
        }
    }
    if ( $obj->{CreationClassName} eq "ServiceSubscriber" ) {
        $subscribers{$obj->{Name}} = 1;
    }
}

## MAIN: open session to Service Assurance Manager
$session = Smarts::session->init( );

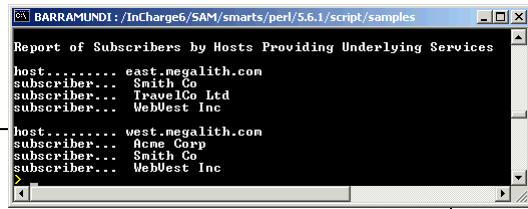
## emit report header
print( "\nReport of Subscribers by Hosts Providing Underlying Services\n" );

## scan all known hosts
foreach my $host ( sort $session->getInstances("Host") ) {

    print( "\nhost..... $host\n" );
    my $hostObj = $session->object( $host );

    ## emit list of all subscribers encountered as object parents
    getParentObj( $hostObj );
    foreach my $subscriber ( sort keys %subscribers ) {
        my $subscriberObj = $session->object( $subscriber );
        print( "subscriber... $subscriberObj->{Name}\n" );
        delete $subscribers{$subscriber};
    }
}

```



Function to find all parents of an object by tracing back known child/parent relationship

Function calls itself to find parent objects of parent objects (recursion)

Add any parents who are ServiceSubscribers to list for this host

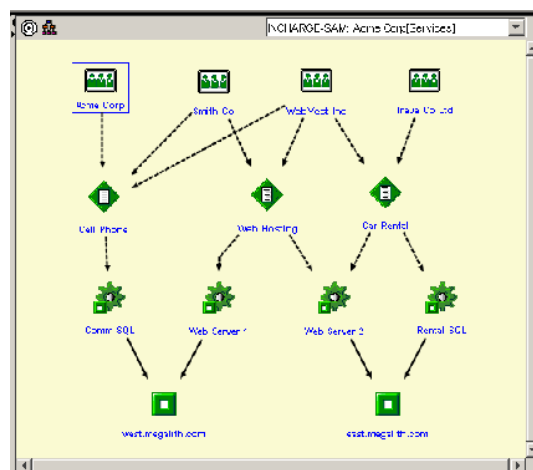
Connect to remote Smarts server

Retrieve list of Managed Hosts and process each

For each host, retrieve and print list of supported ServiceSubscribers

Figure 24: Perl Script for Retrieving ServiceSubscribers Supported by Each Host

Figure 25: ServiceSubscribers by Host Displayed in Service Map



Extending ICIM Library Models to New Domains

The Managed Object Definition Language (MODEL) is a high-level language for describing the attributes, relationships, and behaviors associated with a class of managed entities. The ICIM Library today includes about 100 pre-defined MODEL classes spanning the gamut of objects within infrastructure, application, and business domains, as well as 50 relationships.

The knowledge captured in the ICIM MODEL classes is used to drive the automated analytics of the Smarts platform. With MODEL, developers can easily build new models for new classes of objects by sub-classing and enhancing core ICIM library classes. In this way, developers can extend the reach of Smarts to manage new technology domains that are not covered by off-the-shelf applications.

ICIM classes can represent any logical or physical infrastructure or application objects at any layer, as well as the business objects that rely on them. Similarly, ICIM relationships can represent any type of relationship across infrastructure, application, and business objects, going far beyond the simple connectivity and containment of topology databases offered by most infrastructure management products. ICIM relationships are rich enough to represent the complex web of relationships present in real-world infrastructures including layering, clustering, backup, producer/consumer, client/server, and all others.

Developing models in MODEL offers tremendous advantages. MODEL definitions are abstract, allowing one model to cover a broad set of vendor products. MODEL definitions are declarative, allowing developers to define the key properties of objects, their attributes, operations, relationships, exceptional events, problems, symptoms, and causal propagation, without developing programming logic. And MODEL definitions are compiled, allowing the MODEL compiler to automatically generate all implementation details for objects of the newly defined class. *Most importantly, MODEL definitions drive the Smarts correlation engine, allowing it to recognize exceptional behaviors and authentic problems, and calculate their impacts, all automatically.* Figure 26 shows a simple example of MODEL definitions: the apps.mdl file with three classes: Process, Database, and Server.

```

apps.mdl

interface Process :
    MR_ManagedObject
{
    attribute
        boolean Down = FALSE;
    attribute
        unsigned ProcessID = 0;
    relationshipset
        ServedBy, Database, Serves;
    event
        ProcessDown = Down;
    export
        ProcessDown;
}

interface Database :
    MR_ManagedObject
{
    attribute
        unsigned TransactionCount = 0;
    relationshipset
        Serves, Process, ServedBy;
    relationship
        HostedBy, Server, Hosts;
    problem
        DatabaseDown => ProcessDown;
    propagate symptom
        ProcessDown = Process, Serves;
    export
        DatabaseDown;
}

interface Server:
    MR_ManagedObject
{
    attribute
        string Vendor = "Unknown";
    relationshipset
        Hosts, DataBase, HostedBy;
    problem
        ServerDown => DatabaseDown;
    propagate symptom
        DatabaseDown = DataBase, Hosts;
    export
        ServerDown;
}

```

Figure 26: MODEL Code for Three Classes: Process, Database, and Server

A class has three categories of properties associated with it: attributes, relationships, and events. Figure 27 depicts the three classes showing the potential relationships between the classes, the events associated with the classes, and the propagated symptoms (dotted line).

MODEL captures the generic behavior of a class of objects, separating it from the specific vendor products and the topology of any IT infrastructure. Because the developed model is not tied to any particular products or topology, it can be re-used in any IT infrastructure containing similar types of managed entities. New vendor product support is then added simply by adding new “certifications”, i.e., new mediation layer entries. Figure 28 shows the compilation process for the MODEL file in Figure 26. The MODEL file is compiled by the MODEL compiler into C++ code, which is further compiled using a platform-dependent C++ compiler and linker. The resulting library is loaded into a Smarts server where instances of the classes can be created and events can be monitored.

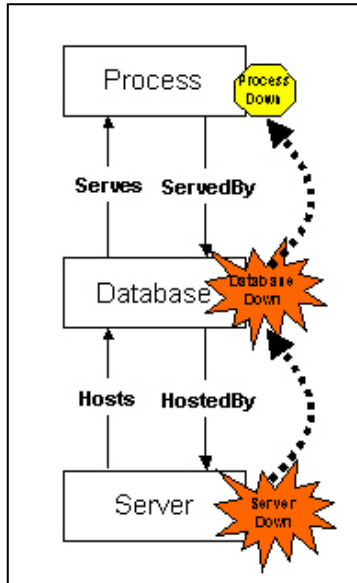


Figure 27: A Visual Representation of the Apps Model

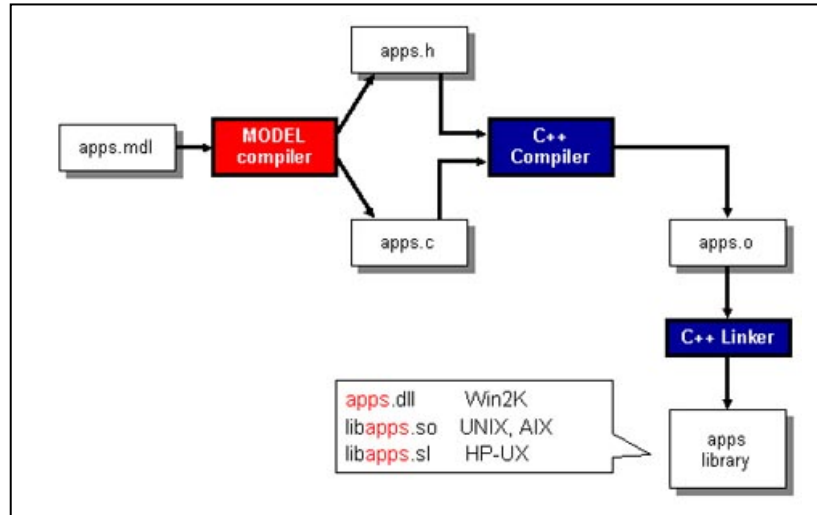


Figure 28: MODEL Compilation Process

Conclusion

Smarts’ unique platform architecture enables powerful analytics to be built into off-the-shelf solutions, resulting in fast deployment, immediate time-to-value, and exceptional ROI. At the same time, Smarts solutions are highly flexible and open, and offer users a variety of ways to both fine-tune the solution to meet custom requirements, and to enhance the rich information in the Smarts Repository to build entirely new management applications.

From the network level to the business level, Smarts offers customization possibilities that have been proven in the marketplace, helping some of the world’s largest companies to maximize the power of Smarts to meet their specific business requirements.

References

1. *The InCharge Common Information Model*, March 2003
2. *Common Information Model: Implementing the Object Model for Enterprise Management*, Bumpus, W. Sweitzer J., Thompson, P., Westerinen, A., Williams, R., John Wiley & Sons, 2000.
3. *The Common Object Request Broker: Architecture and Specification*, Object Management Group and Xopen, 1992.
4. *Common Information Model (CIM) Specification*, Version 2.6, DMTF, June 2002,
5. *Semantic Modeling of Managed Information*, Y. Yemini, A. Dupuy, S. Kliger, S. Yemini, 1993

6. *Event Modeling in the MODEL Language: A Tutorial Introduction*, A. Mayer, S. Kliger, S. Yemini, D. Ohsie, 1997
7. *High-speed and Robust Event Correlation*, 1998
8. *Automating Root Cause Analysis*, 2001
9. *How to Meet Scalability Requirements for Managing the World's Most Complex IT Networks*, 2002
10. *How SMART MoMs Optimize Your Existing Management Investments*, 2002