inWay SA
102, avenue des Champs-Elysées 75008 PARIS
☎+33 1 42 56 41 57 • FAX +33 1 42 89 23 77
http://www.inway.fr • info@inway.fr

intelligence inside

Experience

Active participation in OpenMaster since 1992 • A team of specialists in all fields linked to network management and intelligent systems • Experts in object-oriented software design and development with a genuine on hands experience • A thorough practice and a unique savoir-faire in OpenMaster • A careful consideration of users' needs • A new open-minded approach • A line of products with value added in response to real world expectations.

Innovation

New products in response to real constraints • A strongly structured design • An object oriented architecture based on control/view models • A modular and distributed approach • A constant search for simplicity, efficiency and reusability • An economy of means, an overall coherence, an ergonomic engineering and a finished quality yet unmatched.

Integration

A set of applications and services written with the language and the standard libraries of OpenMaster • A guaranteed compatibility with the existent • Easy communication with any other applications of OpenMaster.

Partnership

An on going dialogue with the users • Clear relationship with the integrators • A constant improvement of the products • A dedicated support of our partners.

steria mummert consulting

Savings

A ready to use solution • Unmatched performance • Reduced resource requirements • One product for multiple purposes • Only one license for one entire project • A price reduction plan on volume.

Offer

inWay's **SmartSuite** for OpenMaster is a set of ready-to-use applications and tools.

Applications

mapper	dynamic graphical view of the network	*explorer*	network/database/process surveillance
logger	alarm logging/consultation/processing	*simulator*	network simulation/control
broker	process communications	*mime*	generic object manager

On the Web

webit	PHP gateway to the Web

MIB Tools

browser	browsing the network	*viewer*	viewing the network
maker	managing the MIB	*notifier*	sending events

Analysis

cstudio	search and definitions of event correlations with time constraints

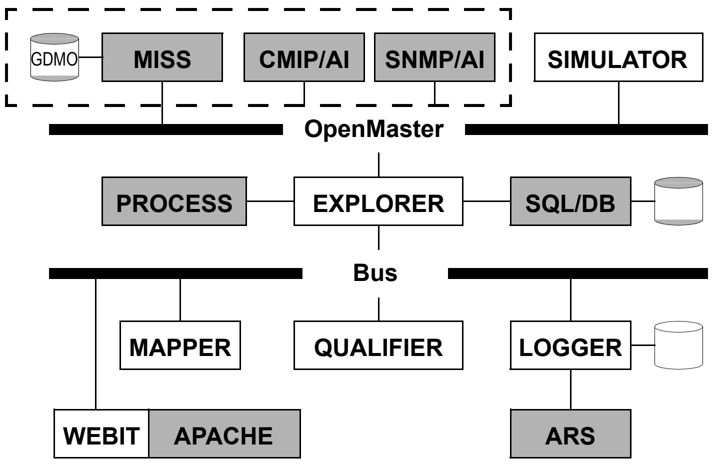
References

inWay's **SmartSuite** for OpenMaster is in operation at the core of many large area network management solutions.

Opportunity

All products by *inWay* are available for a trial period free of charge. Call **Eric Companie** at **+33 (0)3 29 42 03 59** or email at *eric-gilles.companie@inway.fr*.

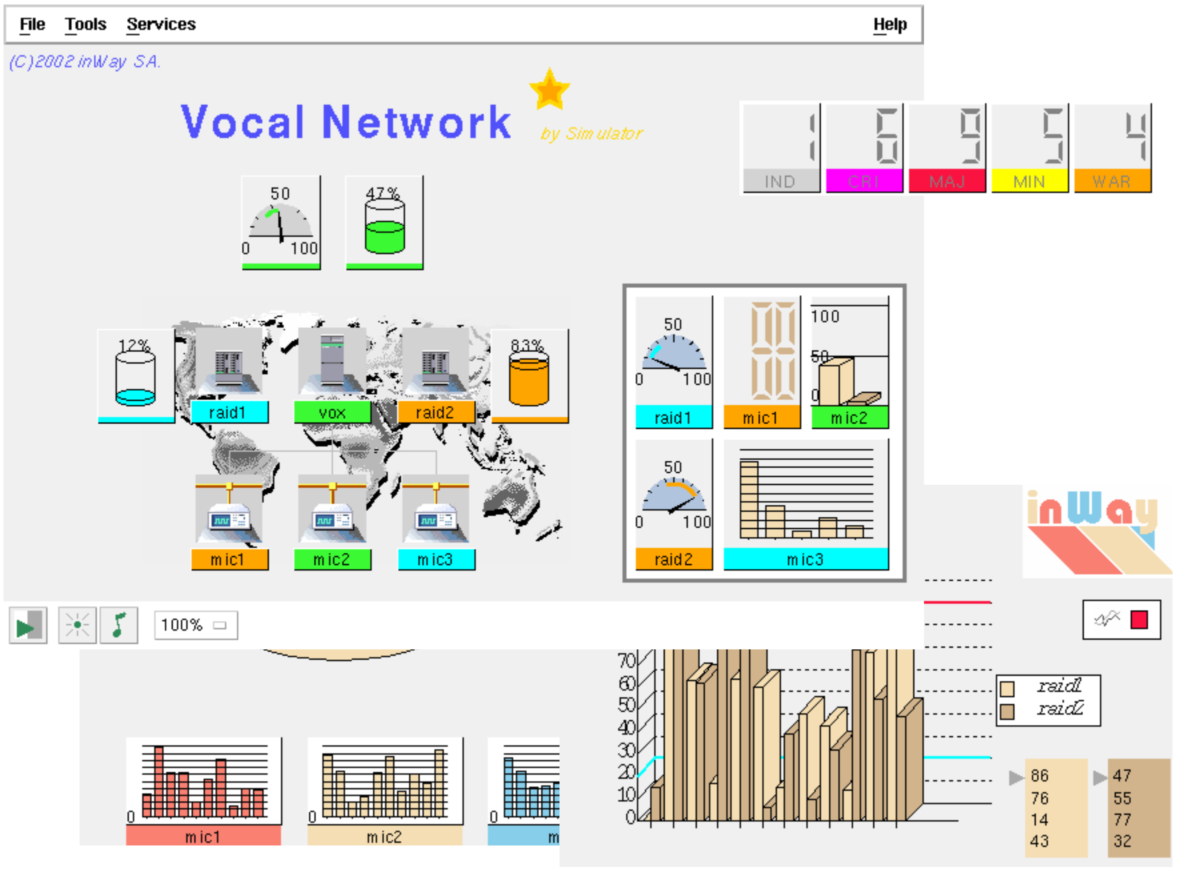
INTEGRATION



- The SMNP and CMIP agent integrators are OpenMaster components which translate operations to network elements and convert event notifications. They are used by the *explorer* to get data and receive events in a normalized form.
- MISS is a standard service offered by OpenMaster which compiles GDMO and RFC documents. MISS can encode and decode information defined in ASN.1/BER and distribute data definitions to other programs. The *explorer* relies on this service to encode and decode information exchanged with network elements. Graphical applications like the editors of the *explorer* and the *simulator* ask MISS for syntax definitions to automatically generate interface objects for network information.
- The *explorer* filters events and watches a MIB, a database or another process for other applications like the *mapper* and the *logger*. It's also a correlation engine. The *explorer* is the only component with the *simulator* which is connected to the OpenMaster framework. Other components like the *mapper* can be connected to the OpenMaster framework if they are extended to act on network elements.
- The combination of agent integrators and MISS from OpenMaster with the *explorer* insures that virtually any network component with an SNMP or CMIP interface can be fully managed by the SmartSuite. If an equipment or an application is lacking a normalized interface, the *explorer* is able to spawn a process and filter its output. This feature can be used to easily watch network elements with a Corba or an ASCII interface.
- The *mapper* shows a graphical representation of the network. It can display simple shapes like polygons and circles, icons, compound objects like histograms and charts. It can draw links between any of these. Maps are animated by changing graphical properties in response to messages sent by the *explorer* or any other process.
- The *qualifier* collects events filtered by the *explorer* and correlates them into alarms which are logged by the *logger*. Its role is to enrich the flow of events, reducing it at the same time. The *qualifier* is embedded directly in the *explorer* or in the *logger*.
- The *logger* keeps alarms and correlated events on disk. Its graphical interface shows alarms, correlated events, history alarms in tables with commands to process them. It includes an open interface to a trouble ticketing system. The *logger* can qualify events on its own or log alarms sent by the *qualifier* or a more sophisticated correlation engine.
- The *simulator* can emulate any network element and act as a command processor on the network. It's a key component to validate and demonstrate solutions, train operators in extreme conditions or act on a network in a controlled way.

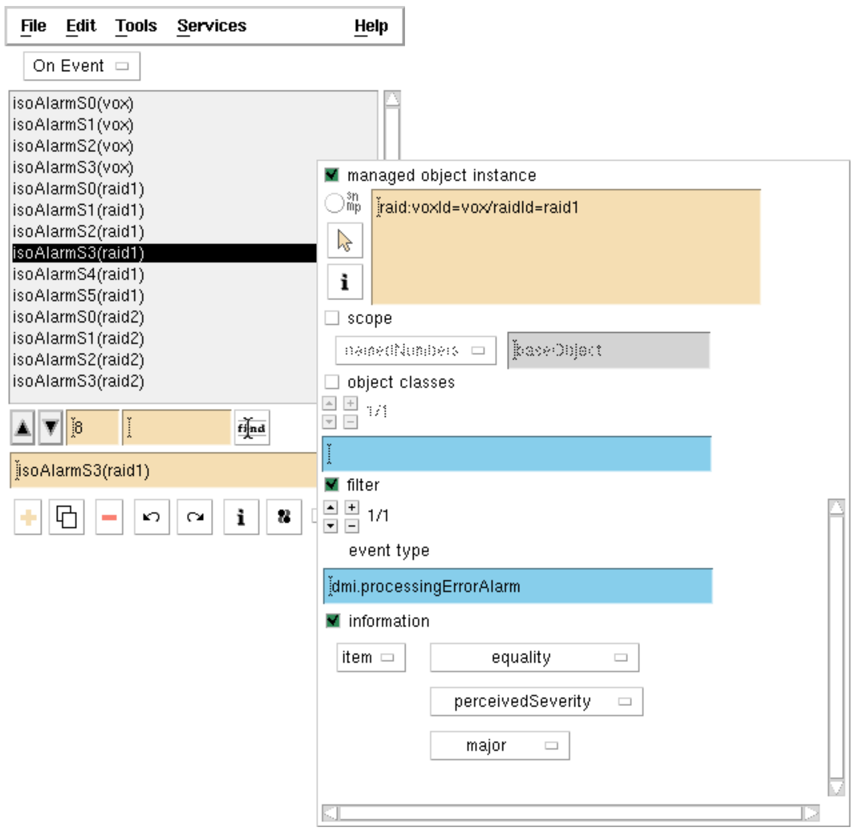
- The *webit* gives complete access to all the information elaborated on the management platform to Web applications. Web technologies like Apache and IIS servers, PHP, HTML and XML formats, XSL and XPATH translators, .NET, Java, JavaScript and Flash can be used freely to exploit all the SmartSuite services.
- All applications communicate transparently via the SML bus with the *broker*. The SML bus is yet another standard component of OpenMaster. SML is the native language of OpenMaster used to write all SmartSuite applications.
- The SmartSuite is amazingly flexible. All components are designed to be as open and as configurable as possible. It’s aimed to give a ready to use solution to integrators with no additional software development.

MAPPER



- shows network elements in graphical maps of animated pictures and links,
- aimed to be the central component in a network and system administration based on OpenMaster,
- the result of much experience in network management and network mapping,
- designed with an object-oriented architecture and around a message-driven control loop,
- can display multiple frames with or without scrollbars, zooming, a main menu and control buttons,
- maps are designed by dragging and dropping objects from palettes,
- graphical properties are changed with dedicated inspectors,
- pictures and links can be animated by any network events and any MIB instances or even a database,
- animation messages change graphical properties with unlimited effects,
- they can also hide, show or start blinking objects,
- moreover, they can trigger personalized sound alerts,
- internal and external actions can be run from contextual menus,
- map files can be generated automatically by program,
- maps can be inserted in specific applications,
- easy to interface with any existing application such as the *logger*,
- delivered with a complete tutorial and a simulation program.

EXPLORER



On Event

isoAlarmS0(vox)
isoAlarmS1(vox)
isoAlarmS2(vox)
isoAlarmS3(vox)
isoAlarmS0(raid1)
isoAlarmS1(raid1)
isoAlarmS2(raid1)
isoAlarmS3(raid1)
isoAlarmS4(raid1)
isoAlarmS5(raid1)
isoAlarmS0(raid2)
isoAlarmS1(raid2)
isoAlarmS2(raid2)
isoAlarmS3(raid2)

8

isoAlarmS3(raid1)

✓ managed object instance: raid:voxId=vox/raidId=raid1

scope: namedNumbers baseObject

object classes 1/1

✓ filter 1/1

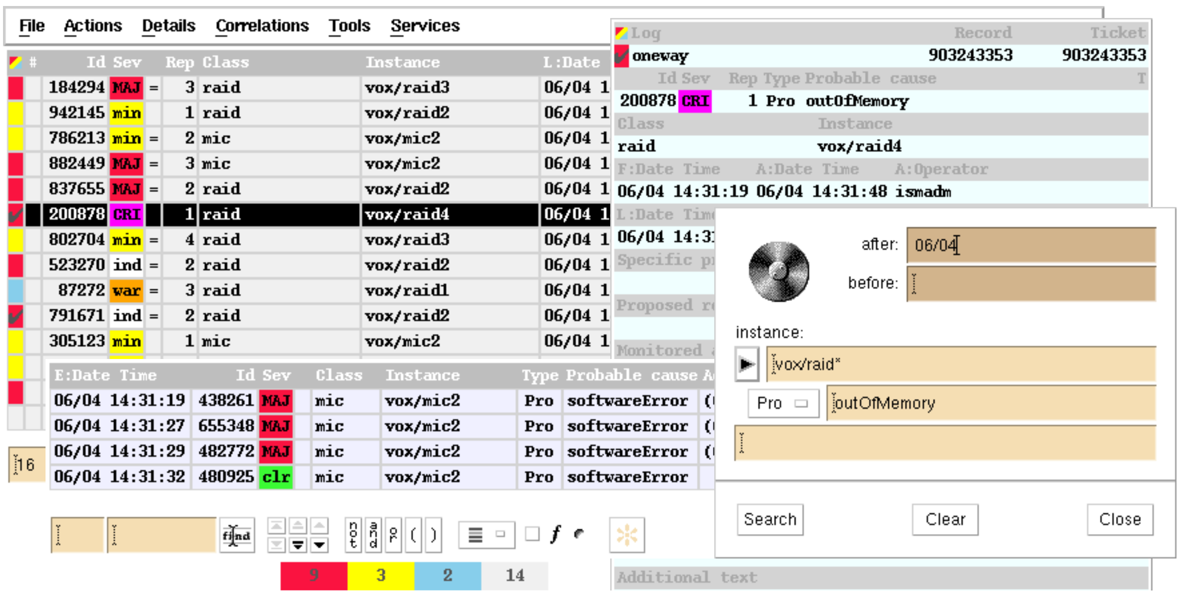
event type: dmi.processingErrorAlarm

✓ information

item | equality | perceivedSeverity | major

- watches a MIB, a database or another process for other applications,
- manages a collection of filters on network events and queries on MIB or database instances,
- awaits for M-EVENT reports,
- periodically runs M-GET or SQL requests and remote function calls,
- sends messages to other applications with the *broker* when an event or an instance is recognized,
- messages can be accompanied with data extracted from an event or a function call, a MIB or a database instance,
- can collect information combining messages and correlate events and queries,
- limits the number of queries on the MIB, a database or another process to minimum,
- particularly designed and programmed to pass filters as fast as possible,
- relieves applications from the complex task of collecting and interpreting management information,
- drastically reduces traffic and processing time both in the framework and the serviced applications,
- can interface OpenMaster for external applications through the *webit*.

LOGGER



#	Id	Sev		Rep	Class	Instance	L:Date
	184294	MAJ	=	3	raid	vox/raid3	06/04 1
	942145	min		1	raid	vox/raid2	06/04 1
	786213	min	=	2	mic	vox/mic2	06/04 1
	882449	MAJ	=	3	mic	vox/mic2	06/04 1
	837655	MAJ	=	2	raid	vox/raid2	06/04 1
	200878	CRI		1	raid	vox/raid4	06/04 1
	802704	min	=	4	raid	vox/raid3	06/04 1
	523270	ind	=	2	raid	vox/raid2	06/04 1
	87272	war	=	3	raid	vox/raid1	06/04 1
	791671	ind	=	2	raid	vox/raid2	06/04 1
	305123	min		1	mic	vox/mic2	06/04 1

E:Date Time	Id	Sev		Class	Instance	Type	Probable cause	A
06/04 14:31:19	438261	MAJ		mic	vox/mic2	Pro	softwareError	(
06/04 14:31:27	655348	MAJ		mic	vox/mic2	Pro	softwareError	(
06/04 14:31:29	482772	MAJ		mic	vox/mic2	Pro	softwareError	(
06/04 14:31:32	480925	clr		mic	vox/mic2	Pro	softwareError	

16

🟥 9	🟨 3	🟦 2	14

Log	Record	Ticket
oneway	903243353	903243353

Id	Sev	Rep	Type	Probable cause	T
200878	CRI	1	Pro	outOfMemory	

Class	Instance
raid	vox/raid4

F:Date Time	A:Date Time	A:Operator
06/04 14:31:19	06/04 14:31:48	ismadm

L:Date Time
06/04 14:31

Specific pr

Proposed re

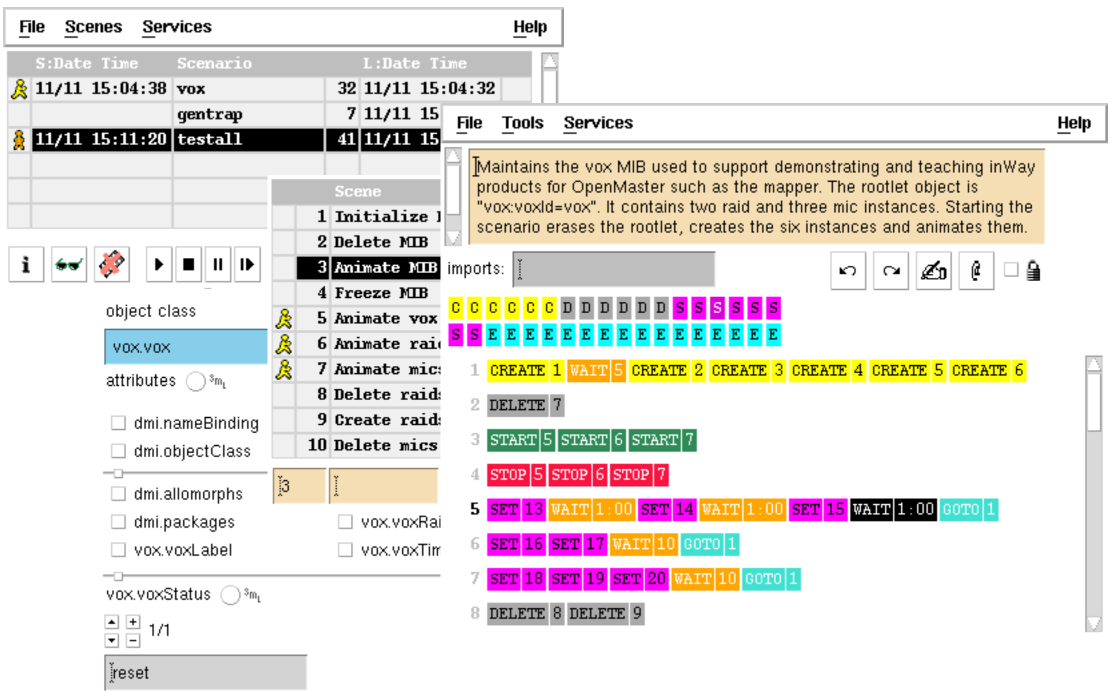
Monitored a

Additional text

after: 06/04
before:
instance:
vox/raid*
Pro
outOfMemory

- displays alarm log records in multiple independent tables with colored text or graphics,
- dialogs with a demon which correlates and logs alarms and events for several concurrent applications,
- controls access to the log with an identification key passed to the demon which fixes what can be seen,
- controls actions on the log with configurable rules,
- reacts immediately to an alarm change, doesn't repeatedly poll the log,
- signals any change by an icon and a personalized sound alert,
- can show all events correlated to a particular alarm,
- can search the history log for passed alarms,
- can display a detail view of any alarm, history alarm or event,
- can open and close trouble tickets,
- can keep displaying terminated alarms for a given time,
- can sort a view by column in ascending or descending order in one click,
- can filter a view on any column or logical combination of columns in one click,
- can search alarms or events with regular expressions,
- can count alarms using programmable filters,
- can print alarms, save alarms in a file and open a previously saved view for off-line processing,
- can move, resize, hide and show columns interactively,
- can be configured in every way depending on different user profiles,
- can be added more columns to display any alarm field or computed data,
- can log and display any type of alarm or event,
- can be extended with more formatting, coloring, counting, filtering and sorting functions,
- can be extended with user functions run from a menu or by double clicking on an alarm,
- can communicate with any other application via the *broker*.

SIMULATOR



	S:Date Time	Scenario		L:Date Time
	11/11 15:04:38	vox	32	11/11 15:04:32
		gentrap	7	11/11 15
	11/11 15:11:20	testall	41	11/11 15

	Scene
1	Initialize I
2	Delete MIB
3	Animate MIB
4	Freeze MIB
5	Animate vox
6	Animate rai
7	Animate mic
8	Delete raid
9	Create raid
10	Delete mics

vox.vox

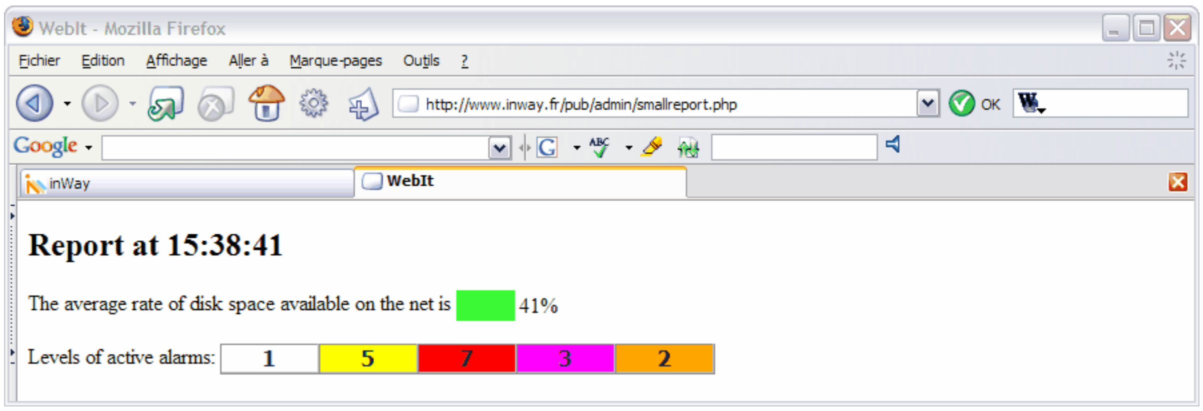
dmi.nameBinding
dmi.objectClass
dmi.allomorphs
dmi.packages
vox.voxLabel
vox.voxRai
vox.voxTim
vox.voxStatus
1/1

Maintains the vox MIB used to support demonstrating and teaching inWay products for OpenMaster such as the mapper. The rootlet object is "vox:voxId=vox". It contains two raid and three mic instances. Starting the scenario erases the rootlet, creates the six instances and animates them.

```
C C C C C C D D D D D D S S S S S S
S S E E E E E E E E E E E E E E E E
1 CREATE 1 WAIT 5 CREATE 2 CREATE 3 CREATE 4 CREATE 5 CREATE 6
2 DELETE 7
3 START 5 START 6 START 7
4 STOP 5 STOP 6 STOP 7
5 SET 13 WAIT 1:00 SET 14 WAIT 1:00 SET 15 WAIT 1:00 GOTO 1
6 SET 16 SET 17 WAIT 10 GOTO 1
7 SET 18 SET 19 SET 20 WAIT 10 GOTO 1
8 DELETE 8 DELETE 9
```

- can emulate network elements or act on real systems,
- can be seen as an object manager or as a command processor,
- provides network actions for creating, deleting and setting instances (M-CREATE, M-DELETE, M-SET),
- can retrieve particular instances (M-GET) and pass the returned attributes to other actions,
- can also send any kind of actions (M-ACTION), events (M-EVENT) or SNMP traps,
- combines actions into scenes where individual actions are run one after the other,
- groups related scenes into individual scenarios saved in separate text files,
- includes processing actions which, in a scene, can start and stop other scenes, add delays between actions, break its normal sequential flow with IF and GOTO statements, make it loop, pass context values between actions, run any SML code and even communicate with other applications using the *broker*,
- consists of three applications: the demon, the runner and the editor,
- the demon runs simulations in the background,
- it can run several scenarios and several concurrent scenes in the same scenario at the same time,
- the runner dialogs with the demon to display the scenarios which are currently loaded, their description and their running and trace conditions,
- it can load and unload scenarios, start and stop them, run individual scenes,
- the editor takes care of building scenarios, scenes and actions,
- scenes in a scenario are presented as a storyboard with different colors for each type of action,
- operations on the MIB, actions and notifications or traps are entered through dedicated inspectors which are automatically generated from GDMO and RFC definitions.

WEBIT

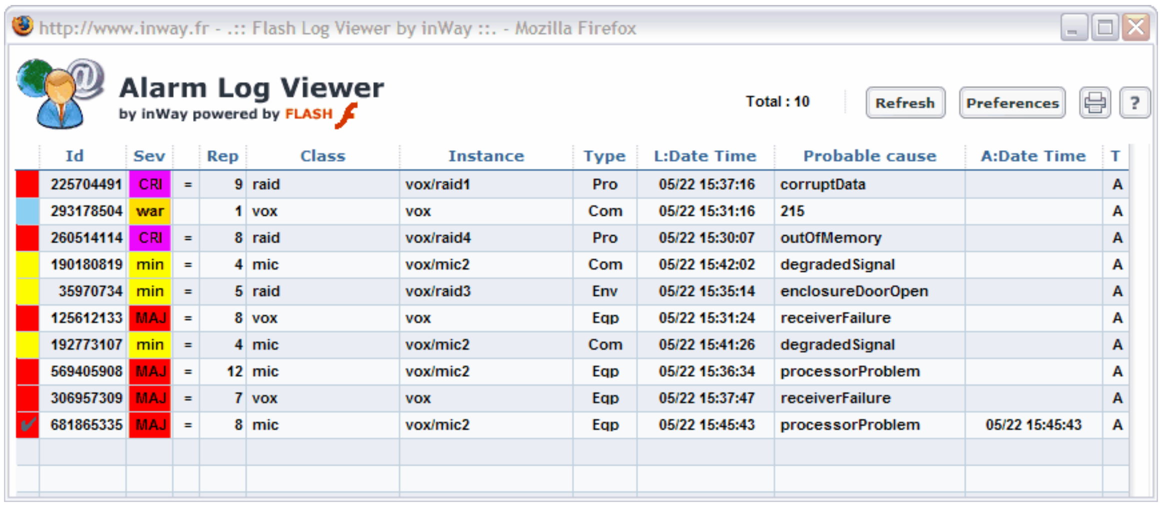

WebIt - Mozilla Firefox
Fichier Edition Affichage Aller à Marque-pages Outils ?
http://www.inway.fr/pub/admin/smallreport.php
OK
Google
inWay
WebIt

Report at 15:38:41

The average rate of disk space available on the net is 🟩 41%

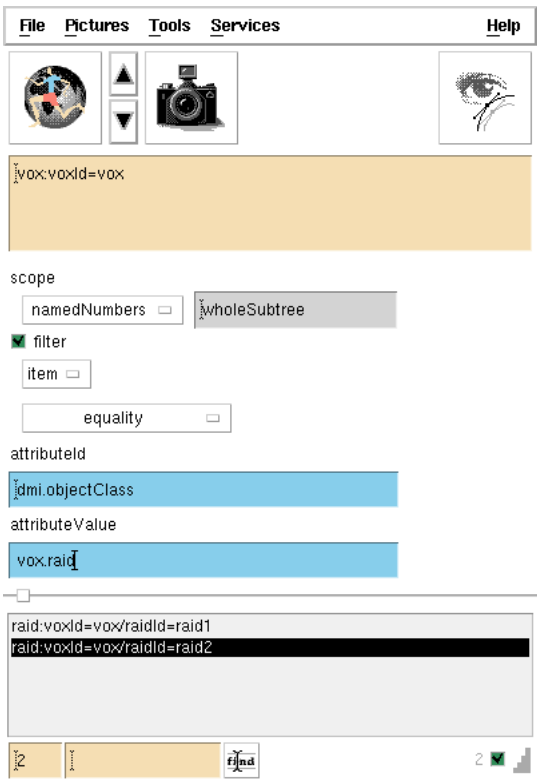
Levels of active alarms:	1	5	7	3	2

- gives complete access in PHP via the broker to all SmartSuite services and applications,
- can dynamically and transparently make available on the Web, via an Apache or an IIS server, in a secure way and without any deployment, all the information elaborated on the network administration platform,
- adds another amazing dimension to the SmartSuite with the possibility to freely use all the technologies found on the web like Apache and IIS servers, PHP, HTML and XML formats, XSL and XPATH translators, .NET, Java, JavaScript and Flash,
- allows web developers to make the most of the power of the SmartSuite and express their skills in producing a vast set of applications going from the simple generation of documents in HTML to plugins in Java or Flash analysing an XML flow.


http://www.inway.fr - .:: Flash Log Viewer by inWay ::. - Mozilla Firefox
Alarm Log Viewer
by inWay powered by FLASH
Total : 10
Refresh
Preferences
?

	Id	Sev		Rep	Class	Instance	Type	L:Date Time	Probable cause	A:Date Time	T
🟥	225704491	CRI	=	9	raid	vox/raid1	Pro	05/22 15:37:16	corruptData		A
🟦	293178504	war		1	vox	vox	Com	05/22 15:31:16	215		A
🟥	260514114	CRI	=	8	raid	vox/raid4	Pro	05/22 15:30:07	outOfMemory		A
🟨	190180819	min	=	4	mic	vox/mic2	Com	05/22 15:42:02	degradedSignal		A
🟨	35970734	min	=	5	raid	vox/raid3	Env	05/22 15:35:14	enclosureDoorOpen		A
🟥	125612133	MAJ	=	8	vox	vox	Eqp	05/22 15:31:24	receiverFailure		A
🟨	192773107	min	=	4	mic	vox/mic2	Com	05/22 15:41:26	degradedSignal		A
🟥	569405908	MAJ	=	12	mic	vox/mic2	Eqp	05/22 15:36:34	processorProblem		A
🟥	306957309	MAJ	=	7	vox	vox	Eqp	05/22 15:37:47	receiverFailure		A
✓	681865335	MAJ	=	8	mic	vox/mic2	Eqp	05/22 15:45:43	processorProblem	05/22 15:45:43	A

BROWSER



vox:voxId=vox

scope

namedNumbers

wholeSubtree

filter

item

equality

attributeId

dmi.objectClass

attributeValue

vox.raid

raid:voxId=vox/raidId=raid1

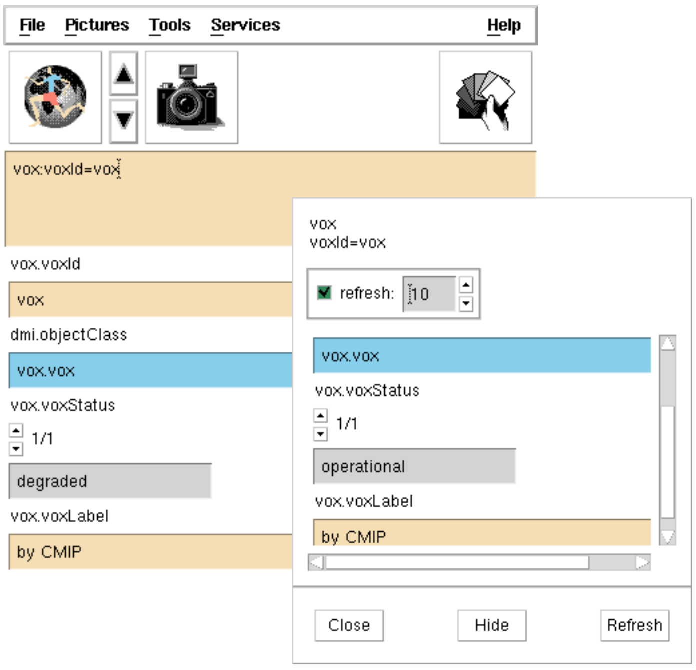
raid:voxId=vox/raidId=raid2

2

2

- navigates in the MIB,
- fully supports scoping and filtering,
- entering specific attribut values in the filter is done with dedicated inspectors generated dynamically from the GDMO,
- can search a long list of instance names with a regular expression,
- can run several requests at the same time in separate frames,
- can save requests in a file,
- specially designed to work with the *viewer*,
- in one click, the *viewer* can import an instance name from the *browser*,
- in one click, the *browser* can ask the *viewer* to display a selected instance,
- as a service via the *broker*, it can be used by any other application to edit and run requests to retrieve instance names.

VIEWER



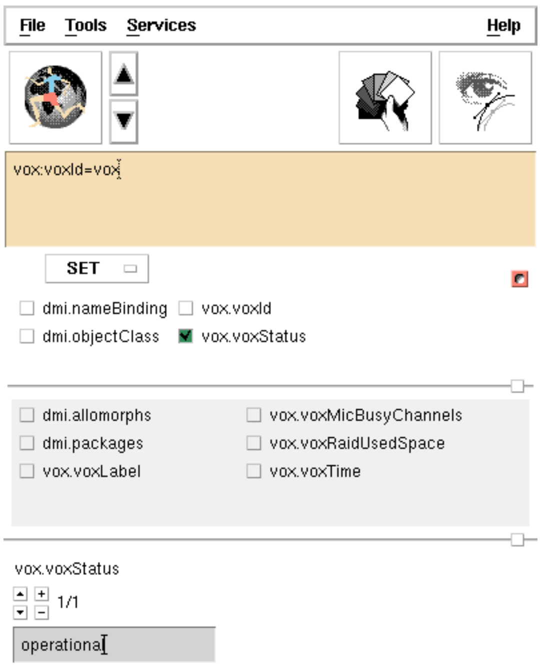
vox:voxId=vox

vox.voxId
vox
dmi.objectClass
vox.vox
vox.voxStatus
1/1
degraded
vox.voxLabel
by CMIP

vox
voxId=vox
refresh: 10
vox.vox
vox.voxStatus
1/1
operational
vox.voxLabel
by CMIP

- displays a detail view of a MIB instance,
- can show any instance,
- attribute values are formatted and displayed with dedicated inspectors generated dynamically from the GDMO,
- can display several instances at the same time in separate frames,
- can update its display automatically,
- can save instance views in a file,
- specially designed to work with the *browser*,
- in one click, the *browser* can ask the *viewer* to display a selected instance,
- in one click, the *viewer* can import an instance name from the *browser*,
- as a service via the *broker*, it can be used by any other application to display MIB instances.

MAKER



vox:voxId=vox

- [ ] dmi.nameBinding
- [ ] vox.voxId
- [ ] dmi.objectClass
- [x] vox.voxStatus

- [ ] dmi.allomorphs
- [ ] vox.voxMicBusyChannels
- [ ] dmi.packages
- [ ] vox.voxRaidUsedSpace
- [ ] vox.voxLabel
- [ ] vox.voxTime

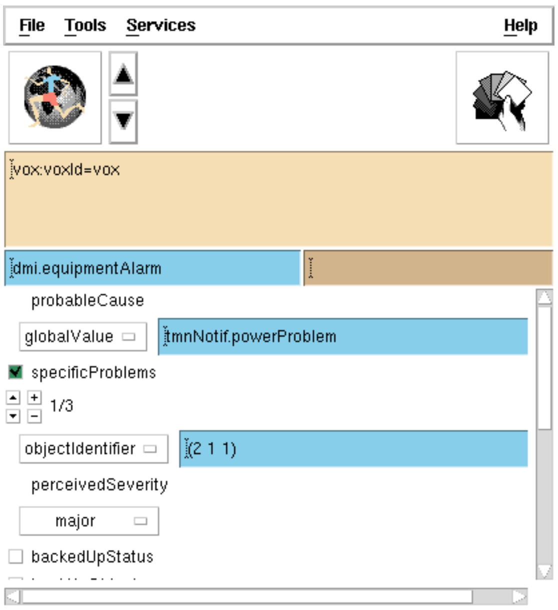
vox.voxStatus

1/1

operational

- creates, deletes and sets MIB instances,
- can operate on any instance,
- attribute values are entered with dedicated inspectors generated dynamically from the GDMO,
- can act on several instances at the same time in separate frames,
- can save operations in a file,
- specially designed to work with the *browser* and the *viewer*,
- in one click, the *maker* can import an instance name from the *browser*,
- in one click, the *maker* can ask the *viewer* to display the targeted instance,
- as a service via the *broker*, it can be used by any other application to edit and send operations on the MIB.

NOTIFIER



vox:voxId=vox

dmi.equipmentAlarm

probableCause

globalValue | tmnNotif.powerProblem

specificProblems

1/3

objectIdentifier | (2 1 1)

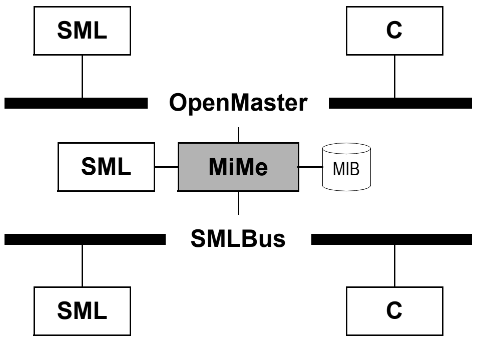
perceivedSeverity

major

backedUpStatus

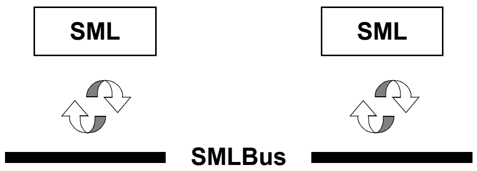
- reports an event,
- can edit and send any event,
- parameter values are entered with dedicated inspectors generated dynamically from the GDMO,
- can edit several events at the same time in separate frames,
- can save events in a file,
- specially designed to work with the *browser*,
- in one click, the *notifier* can import an instance name from the *browser*,
- as a service via the *broker*, it can be used by any other application to edit and send events.

MIME



- maintains a MIB in memory with CMIS primitives,
- reads thousands of instances from disk in seconds,
- behaves exactly like an object manager for the framework,
- fully supports CMIP,
- can easily be extended in order to build a specific application,
- can be used to write alarm logs, proxy agents, virtual MIB managers, etc.,
- fully compatible with SML CMIS,
- answers also to requests directly sent in SML via the *broker*,
- enables applications to talk CMIP without using the communications framework.

BROKER

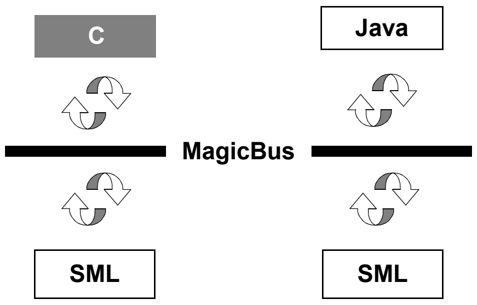


- turns any SML application into a distributed service without any extra code on the server or the client side,
- connections, remote function calls, argument passing and error returns are made transparent,
- any error on the server side is captured and reported to the client as if it happened locally in its own process,
- interprets a dialog in SML, in XML-RPC and in serialized PHP,
- the *broker* radically extends the possibilities of implementations by offering to share roles without any constraint.

```
;; in an SML process, define a function in a normal way
? (defun calc:add (x y) (+ x y))
;; export it and register on the bus
? (load "smlb")                ;load broker
? (smlb:export 'calc:add)      ;function is now public
? (smlb:start "Calculator")    ;register on the bus

;; now in another SML process
? (load "smlb")                ;load broker
;; import function and simply run it
? (smlb:import "Calculator" 'calc:add)
? (calc:add 1 2)               ;remote call with automatic connection
= 3
? (calc:add 1)                 ;errors are returned as if local
Error: Bad number of arguments: 1
```

MAGICBUS



- exchanges data between separate processes over a TCP/IP channel,
- processes can enter and leave the *magicbus* whenever they want to,
- a process can register itself and change its name on the *magicbus* at any time,
- only one connection channel is needed to communicate with several other processes,
- the same uniform interface can be used to exchange data between different applications,
- communication with the *magicbus* is asynchronous,
- a process can be a server and a client at the same time,
- a message can be a notification and no reply is expected,
- a message can be a request in which case a reply is expected,
- if a reply isn't received after a given fixed time, an error is automatically reported,
- a message can be broadcast to several processes registered under the same name,
- when a given process fails, messages are automatically redirected to a process with the same name,
- there is no limitation to the number of communicating processes,
- there is no limitation to the size of the messages processes can exchange,
- a missing peer process or a transmission error is always reported,
- can be interfaced in any programming language.