Monitoring Design (Conceptual Solution)

SDF Monitoring

Document Information

General	
Document Title:	Monitoring Design (Conceptual Solution)
Area or Project:	SDF Monitoring
Document Code:	
Document Owner:	Simon Rosman
File Name	Monitoring Design (Conceptual Solution).docx

History					
Version	Date	Status	Author	Affected pages:	Approved by:
V0.1	22-09-2009	New	Simon Rosman	All	
V1.0	28-09-2009	Concept	Simon Rosman	All	
V1.1	12-10-2009	Update	Simon Rosman	All	
V1.2	13-10-2009	Final	Simon Rosman	All	Gert Smits, Wijnand van Plaggenhoef

Distribution			
Version	Name	Date	Publication Status

TABLE OF CONTENT

1. Introduction

1.1 Content

This document describes the monitoring of the Cordys multi-cluster environment. Topics dealt with are:

- The current situation
- Problems of the current situation
- Functional requirements for monitoring
- Non functional requirements for monitoring
- The conceptual monitoring solution
    - Integration scenarios (Cordys < -> Monitoring tool)
    - Required changes in SDF
    - Procedures for monitoring the Cordys components

1.2 Audience

The intended audiences are:

- SaaS Deployment Framework team
- Operational management team

1.3 Purpose of this document

This document serves a number of purposes, namely:

- To put a first stake in the ground. Until this moment there is no document describing monitoring of a Cordys multi-cluster environment.
- To discuss what the minimum requirements are for a monitoring setup.
- To discuss what kind of architecture will be used for monitoring the Cordys multi-cluster environment.

1.4 Definitions, acronyms, and abbreviations

Term	Description
SaaS	Software as a Service
Cordys BPMS	Business Process Management Suite
Cordys Cluster	A group of tightly coupled servers with Cordys BPMS installed.
Provisioning	The process of remotely distributing and managing applications, organizations, users and all related data.
Metering	The process of logging the usage of applications and services.
SDF	SaaS Deployment Framework The Cordys product used for provisioning and metering across a number of clusters.
JMX	Java Management Extensions JMX enables a Java application to be remotely monitored.
JMX counter	This document uses this term as follows: A JMX component, placed in an application, which allows external applications to retrieve information, such as status, from a Java application.
OSI Model	Open System Interconnection Reference Model A description of computer communication in 7 layers. Layer 7 is abstract, layer 1 is the simplest.

2. Environment

2.1 Project history

In the past, a similar project about monitoring has been run. However, the project had a different focus. The project was about monitoring Cordys installations from customers. Customers with their own Clusters wanted to monitor their Cordys installations with their existing monitoring tool(s). (See the Cordys wiki at: https://wiki.cordys.com/display/dsc/How+ to+ manage+ Cordys+ with+ Nagios+ monitoring + software)

Monitoring dashboard

This project differs from the previous project because this is about monitoring Cordys' own situation. It is also different because the entire Cordys multi-cluster environment has to be monitored instead of a single cluster.

2.2 Related projects

This project is related to several other projects.

- The project described in " 2.1 Project history"
- The Cordys Google project
  Cordys has started a project with Google to open an application store and various online business solution products. The monitoring solution is created keeping in mind the Cordys + Google environment requirements.

2.3 Technical Environment

2.3.1 Standard Cordys Setup

Basic knowledge of the Cordys environment is required in understanding the project. Therefore this document will give a brief explanation of the Cordys architecture.

Cordys uses a standardized environment for running the Cordys BPMS. This environment is shown in the following picture.

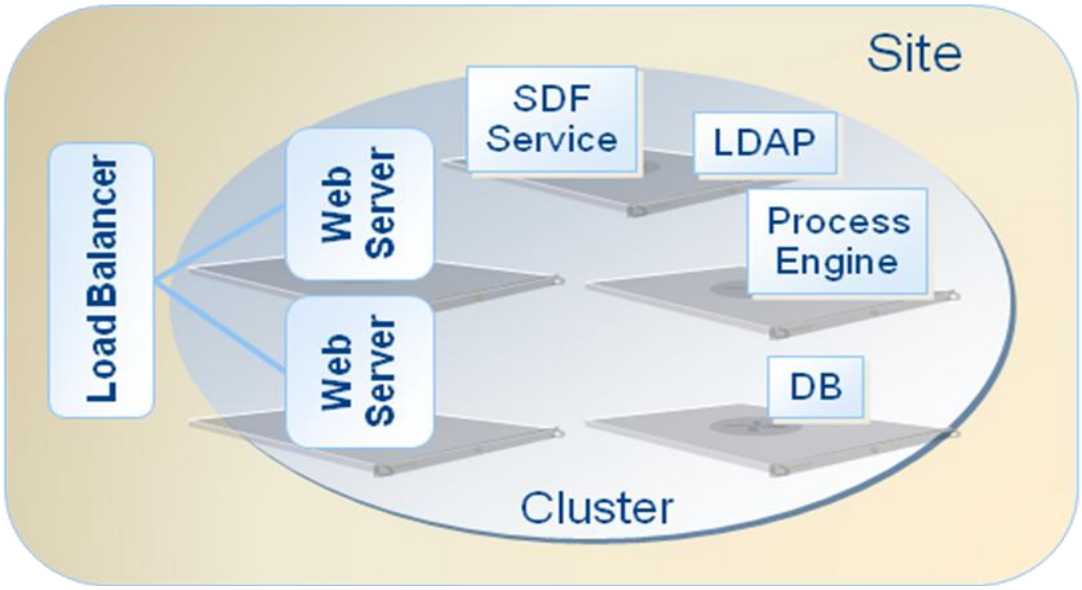


Figure 2.3.1 A standard Cordys cluster

This formation is called a Cluster. Cordys has deployed a Cluster intended for Customers only. This is called a Customer Cluster. Cordys has another cluster, at a different location, which is build for managing Customer Clusters, which is called (the) Admin Cluster. The entire environment is shown in the following picture.

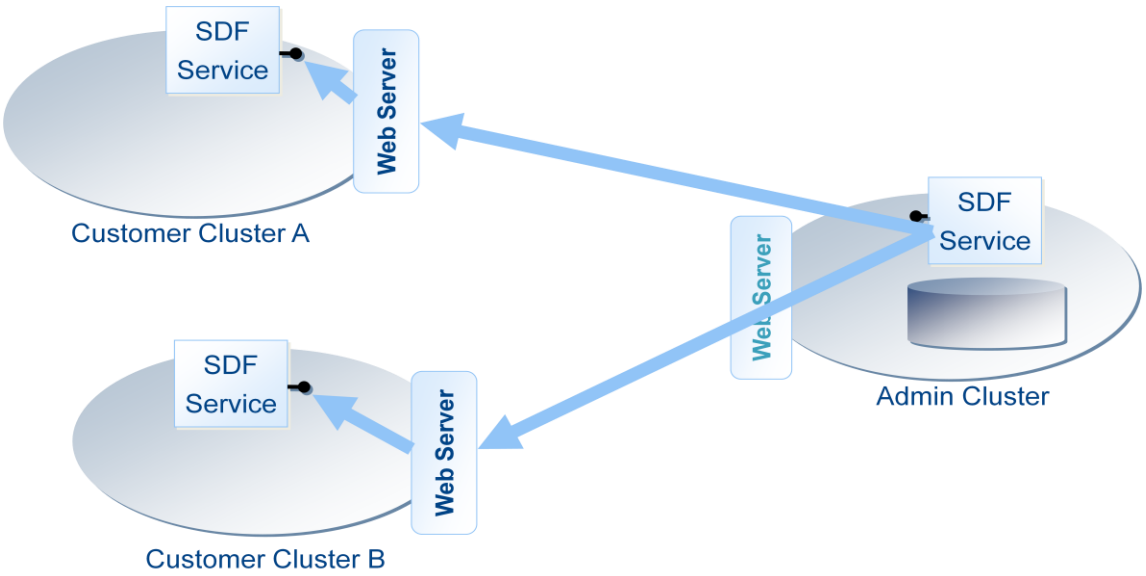


Figure 2.3.2 A Cordys multi-cluster environment

2.3.2 Provisioning

SDF is the Cordys component which provides provisioning across clusters. SDF supports provisioning of the different items.

Organization Provisioning
Organization provisioning will determine the cluster on which the organization will be created and will than create the organization on the chosen cluster. The SDF bookkeeping on the Admin cluster is updated with the new organization.

Application Provisioning
A customer can request the use of applications and the application provisioning is the process of making the application available within the organization of the customer. Part of the application provisioning is the process of deploying and configuring the application on

the given cluster. This is in SDF 3.0 a manual process. The SDF bookkeeping on the Admin cluster is updated with the information that the application is available to the organization.

User Provisioning

User provisioning is the process of creating a user in a given organization. User provisioning is done within the context of an organization and is started with a user registration request (self service) or when an administrator creates the user. In the provisioning process, the cluster of the organization is determined and the authenticated user and organization user are created. The SDF bookkeeping on the Admin cluster is updated with the information that the user is added to the organization. The authenticated user ID is in all clusters and the SDF bookkeeping equal.

3. Problem

3.1 Problem description

Whenever a problem arises in one of the provisioning tasks, or the supporting infrastructure, finding the source of the problem is usually taking a long time. The Cordys cluster does monitor itself, but not the supporting components, such as server hardware, network components and connections. For example, when a web server is disabled, it results in malfunctioning provisioning, but the cause is not visible. This is because Cordys was not designed with multi-cluster in mind, so most facilities are local. Therefore also inter-cluster problems are not dealt with.
The applications keep log of events and errors, but logs are not stored at a central location, and they are in various different formats, XML, Plain text and in databases. Some of the logs files are too large to be useful.
This problem affects operational management.

> In short: Too little information is available of the hardware, network and the infrastructure, and too much (or too distinct) information of the SDF to manage the entire environment, and to be able to solve problems adequately.

3.2 Purpose

The purpose of the project is to create a solution which enables Operational Management to overview & monitor the status of the entire Cordys multi-cluster environment. The solution must enable administrators to investigate and act quickly in case of problems.

3.3 Job

The job is to provide a solution which corresponds to the projects purpose.
This solution must:

- Provide a single point from which administrators can overview the Cordys multi-cluster environment, to be referred to as ‘ Error cockpit’ .
- Provide administrators with up-to-date information about the Cordys multi-cluster environment, i.e. monitoring.

As part of this solution the following tasks must be done:

- Listing requirements and creating a design for a new monitoring environment, to be referred to as (the) monitoring design.
- Monitoring a default installation of a Cordys environment, including software and hardware, using an independent monitoring tool
- Expanding SDF with extra JMX counters or monitoring existing JMX counters
- Integrate the external monitoring tool and the Error Cockpit.
- Checking and improving the SDF structure regarding monitoring.

4. Monitoring principles

4.1 The concept of monitoring

In this document, the following concept of monitoring is used: Monitoring is done using external components which are not part of the ‘to be monitored’ environment. This is shown in the figure below.

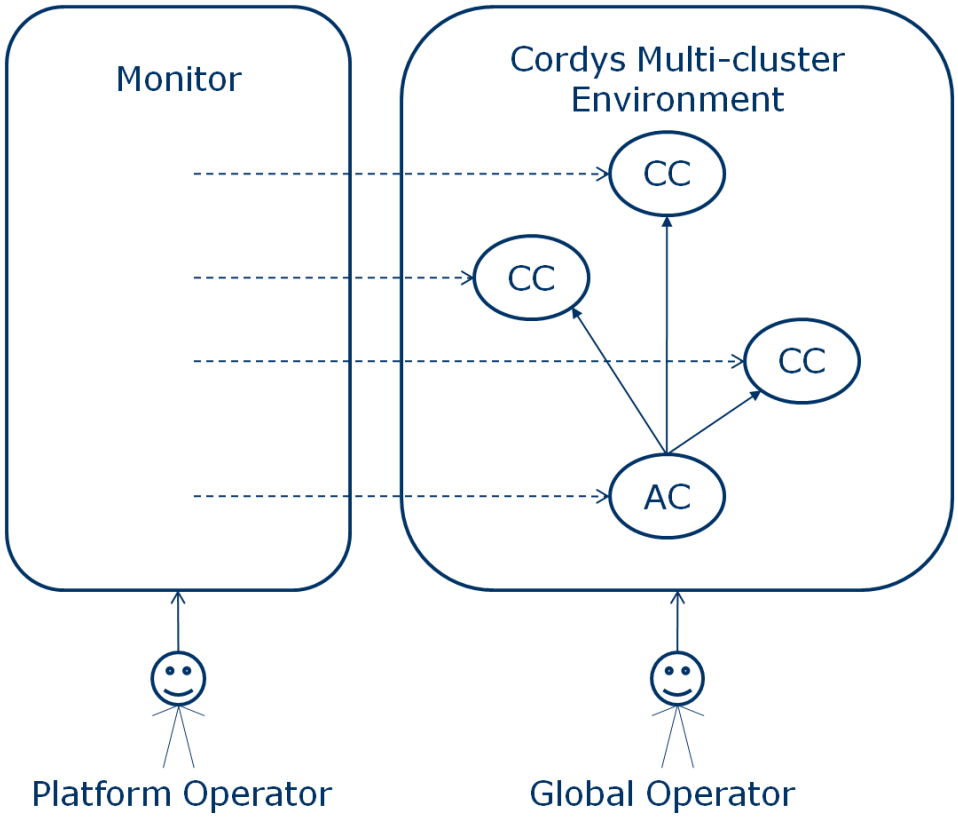


Figure 4.1.1 Monitoring is independent of monitored objects. (AC: Admin Cluster, CC: Customer Cluster)

Using this concept, it is possible to use the monitor even when the Cordys Multi-cluster Environment is not functioning properly.

4.2 Workflow for error handling

This document assumes problem handling is done according to the following workflow. This workflow describes what must happen in case of errors. This workflow is initiated by a monitoring tool, which is continuously monitoring the Cordys Multi-cluster environment.

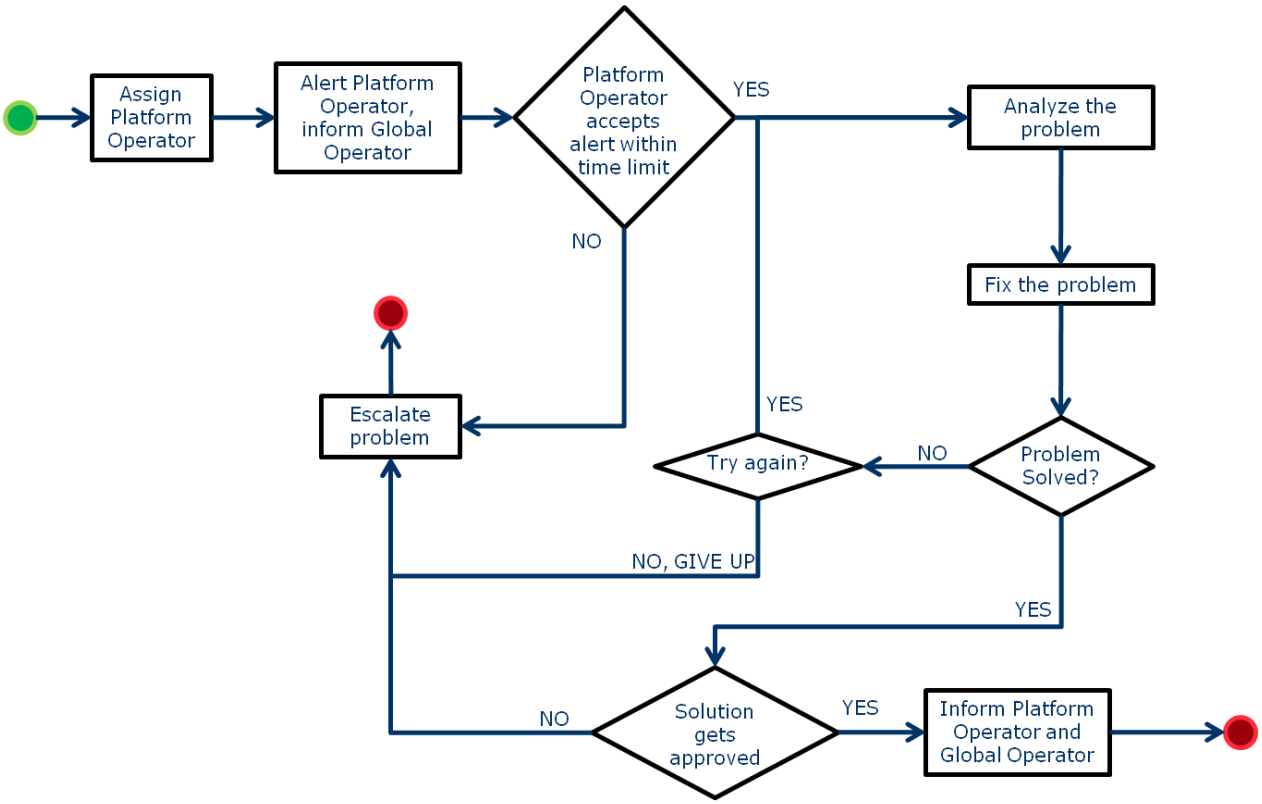


Figure 4.2.1 Workflow for error handling

4.3 Definition of a Monitoring tool

In this document, the following symbol is used to represent a monitoring tool:

Figure 4.3.1 Monitoring tool symbol

In this document, a monitoring tool consists of the following:

1. The core, which monitors the desired targets, generates alerts, etc.
2. A Notification component to send e-mails and SMS upon alerts.
3. A database or data storage to keep logging history etc.
4. A user interface to provide administrators with information about the monitored environment.
5. Integration component(s), to communicate events to other applications etc. (optional).

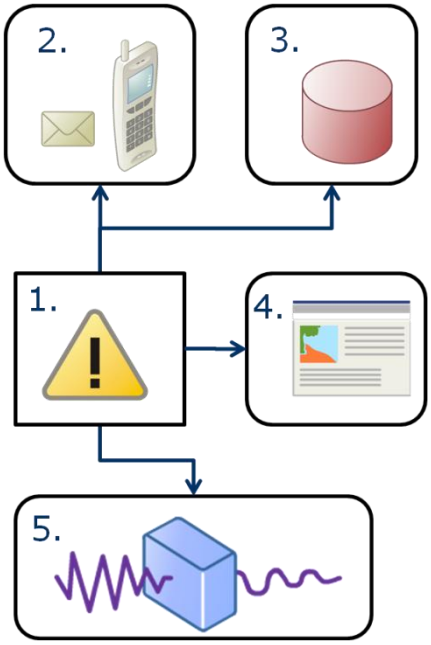


Figure 4.3.2 Monitor components

5. Functional Requirements

5.1 Overview of the entire environment

Administrators must be able to analyze the entire Cordys Multi-Cluster environment in a single overview. This includes admin cluster and customer clusters.

5.2 Pushing alerts in case of problem

Administrators must get alerts in case of any problem.

- This reduces the response time
- This reduces the necessity for manual checking.

5.3 View all information regarding a (potential) problem

Administrators must be able to access all information about (potential) problems using one tool. This tool must provide a list of the tools available for problem analysis.

This information must be available:

- When did the problem occur?
- Where did it occur?
  - Which cluster
  - Which SOAP Processor / machine
- Why did it occur?
  This information is not always directly available, but there must be related information available such as:
  - What component did cause the problem
  - Are there any related errors
  - Who caused the problem?
    - A user
    - A system
  - What kind of problem is it?
    E.g. was it caused by a time-out or a failure?

5.4 Gathering knowledge

Administrators must be able to search through knowledge gathered from historical problems and events. Administrators must record their solution when solving a problem, thus gathering knowledge for future usage.

5.5 Information sources

Administrators must be able to access the following information sources in order to analyze (potential) problems and fix or prevent (potential) problems.

5.5.1 Engine Tasks

Administrators must be able to analyze provisioning tasks.

5.5.1.1 Real-time overview of provisioning tasks

Administrators must be able to view changes in engine tasks in real-time.

5.5.1.2 Provisioning state model overview

Administrators must be able to examine the provisioning state model graphically.

5.5.2 SOAP Processes

Administrators must be able to view the state of SOAP processors and manage SOAP processors.

5.5.3 Process Instances

Administrators must be able to view process instances from the host organization and from all other organizations.

5.5.4 LDAP

Administrators must able to check the contents of the LDAP directory for changes regarding a provisioning task.

5.5.5 Cordys Log Files

Administrators must be able to view error information from log files from all clusters.

5.5.6 SQL Databases

Administrators must be able to do the following regarding SQL databases.

5.5.6.1 Database Health

Administrators must be able to view the health of all the SQL databases used in the Cordys multi-cluster environment.

5.5.6.2 Provisioning Run-time BPM

Administrators must be able to check the contents of the SQL databases regarding a run-time BPM of a provisioning task.

5.5.7 Cluster connectivity

Administrators must be able to check on low level connectivity between clusters or cluster machines.

6. Non Functional Requirements

6.1 Extensibility of Architecture

It is not possible to foresee the needs of future customers, which customer specific objects need to be monitored and what is needed to do so. For that reason the architecture needs to provide an extension mechanism to add new sets of objects to be monitored.

6.2 Independency of architecture

The monitoring tool must be completely independent of the Cordys product. This means that Cordys must be able to operate while the monitoring tool is not functioning properly, and the monitoring tool must be able to operate without Cordys running.

6.3 Open communication

The created design must not require a specific monitoring tool. The design parts affecting the SDF must be generic and must allow the use of any monitoring tool.

6.4 User Interface Experience

The extensibility of the product must not lead to a scattered interface where users must use different tools and user interfaces for doing a single task. The user tasks will consist of dialogs that can be composed into a single user interface.

6.5 Error Handling

There needs to be a single view on error handling to provide the operators one place to monitor the system and handle issues. The error message dialog consolidates the messages generated by the different modules and also provides drill down facilities needed to understand the issue.

6.6 Open source software

If a non-Cordys product is chosen to perform monitoring, it must be Open Source.

6.7 No single point of failure

To provide a high available solution to our customers, no single point of failure must exist in the monitoring design.

6.8 Multiple clusters

The number of clusters to be monitored could expand in the future. Therefore, a method to add monitoring to a cluster must be provided. Adding monitoring to a cluster must not require more time than installing the Cordys itself.

7. Conceptual Solution

7.1 Independent monitoring

This is a Cordys cluster environment, showing the objects to be monitored.
The monitor continuously monitors the Cordys Multi-cluster Environment.

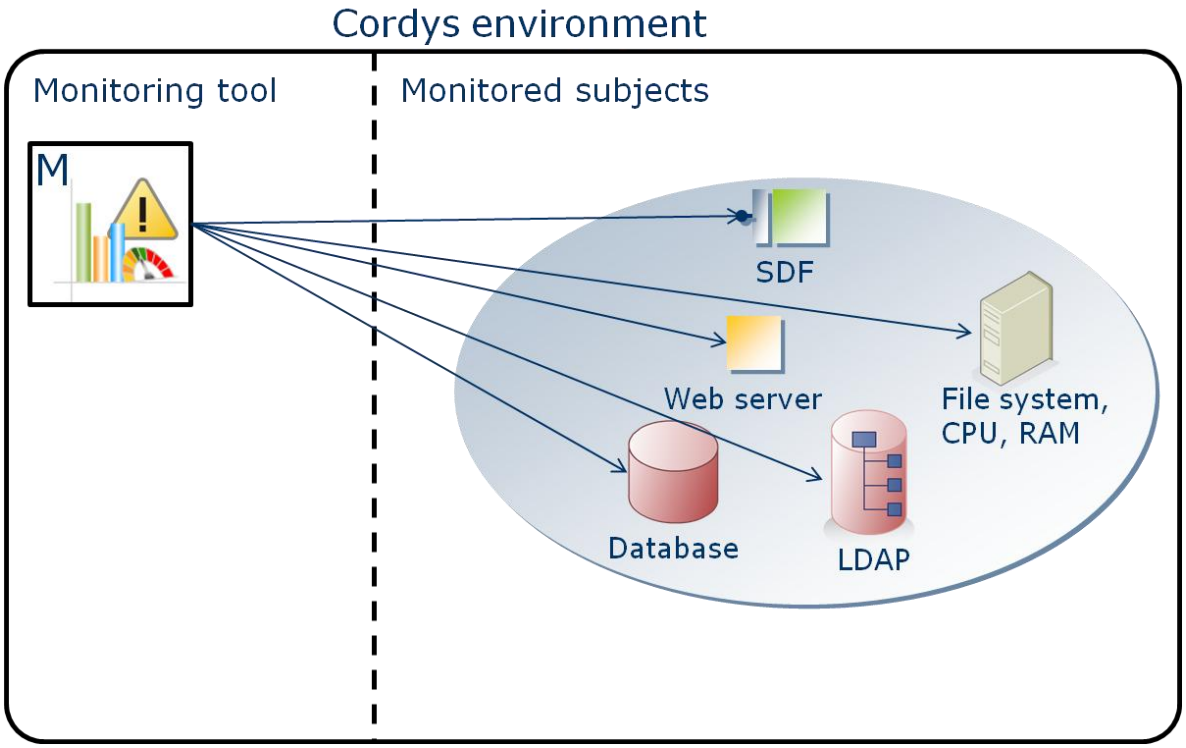


Figure 7.1.1 Cordys Environment

Note: The monitoring tool is NOT part of the Cordys State SyncUp ring, but it could consist of Cordys technology.

7.2 Positioning the monitoring tool in the Cordys architecture

7.2.1 Scenario 1: Master – Slave setup

Preferably, the monitoring tool is inside the Cordys cluster site environment. So the setup will be as follows: Each cluster site has its own monitoring instance. The monitoring slaves send updates to a central master monitoring server.

Pros

- This will allow the tool, to detect low level errors.
- All internal cluster components are directly accessible, without accessing them externally.
- In case of internet connectivity problems, each cluster is still being monitored independently.

Cons

- The Cordys cluster is exposed to vulnerabilities (bugs) in the monitoring tool.
- This risk can be eliminated by limiting external access to the tool.
- Each Cordys cluster needs its own instance of the monitoring tool.

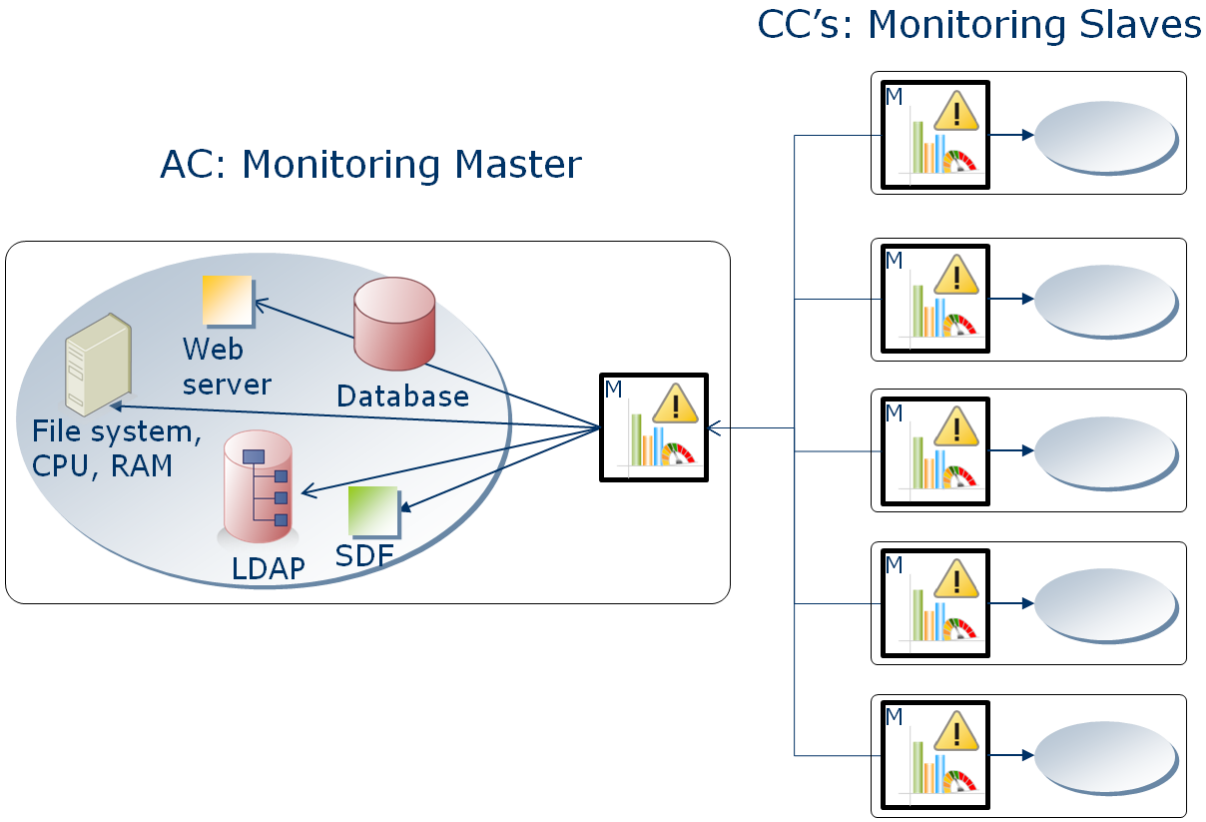


Figure 7.2.1 Master Slave setup

Note: This concept is called distributed monitoring.
This solution is chosen because it best matches the Cordys multi-cluster architecture.

7.2.2 Scenario 2: Remote monitoring

Another solution is to monitor the entire Cordys multi-cluster environment from the outside, for example over the internet.

Pros

- This reduces the number of monitoring instances (failover is necessary).

Cons

- All monitoring data uses the internet connection, so there will be a huge data increase.
- This solution creates security issues because all cluster components must be directly accessible by the external monitoring tool.
- No monitoring in case of connectivity problems.

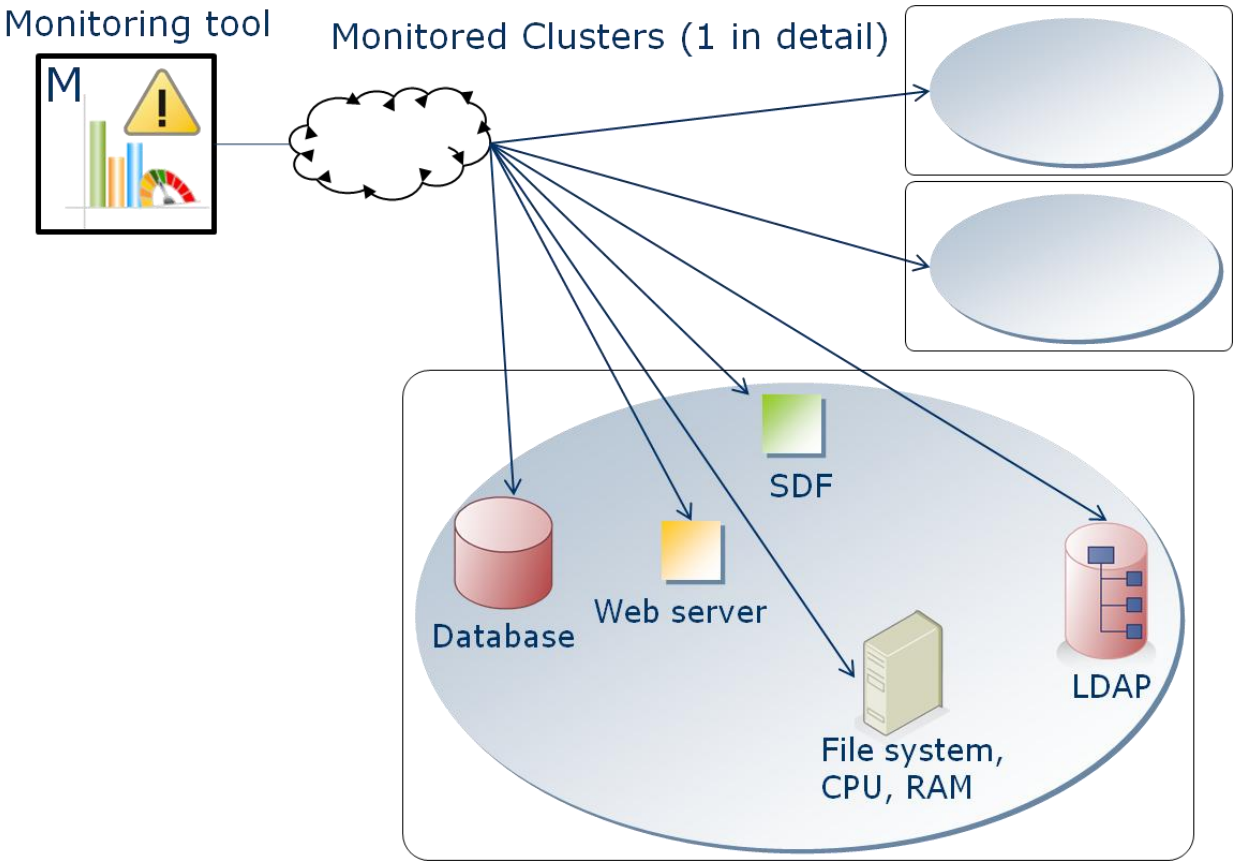


Figure 7.2.2 Monitoring Cordys from outside the network

This document doesn't handle this concept any further, but focuses on the master-slave concept.

7.3 Integrating the monitoring tool and Cordys

The following sections describe different integration scenarios. One of these must be chosen.

7.3.1 Scenario 1: No integration

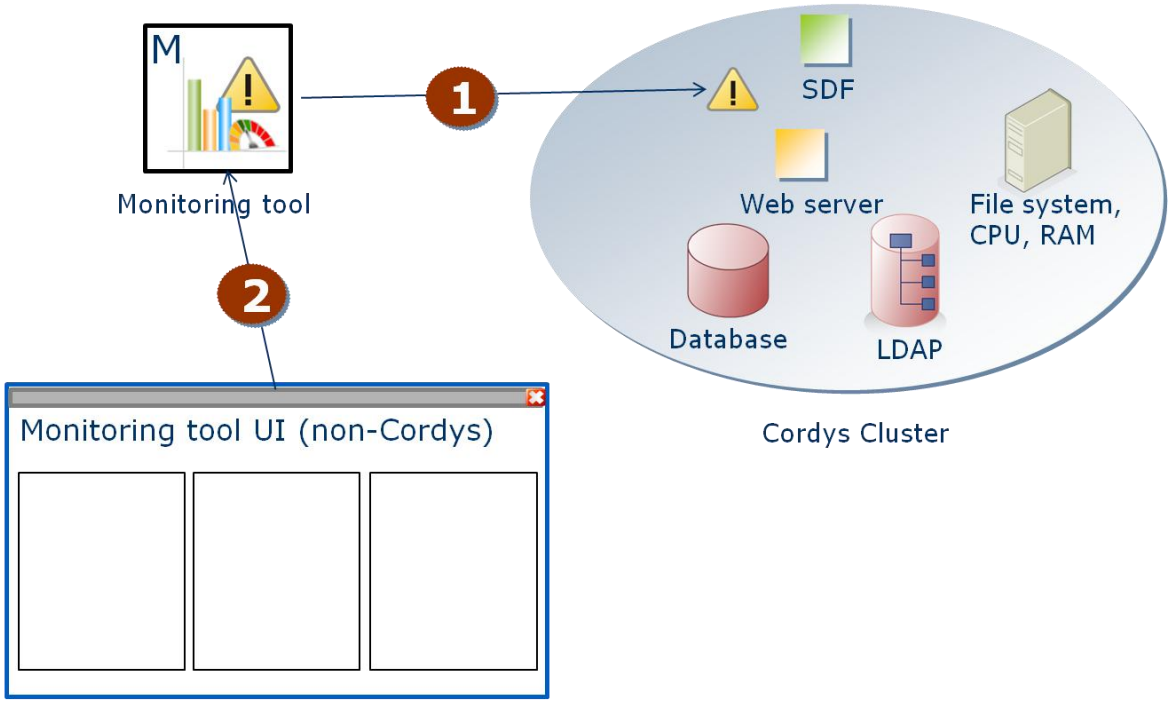


Figure 7.3.1 No integration between Cordys and the monitoring tool

The following steps will are performed

1. The monitoring tool detects a problem.
2. The UI displays only data from the monitoring tool.

Pros

- Easiest solution.
- Completely independent solution.

Cons

- Only information from the monitoring database is available in the monitoring UI.
- Using Cordys components, such as the engine tasks monitor, is not possible.
- All necessary Cordys logic must be build into the monitoring tool.

7.3.2 Scenario 2: Data integration

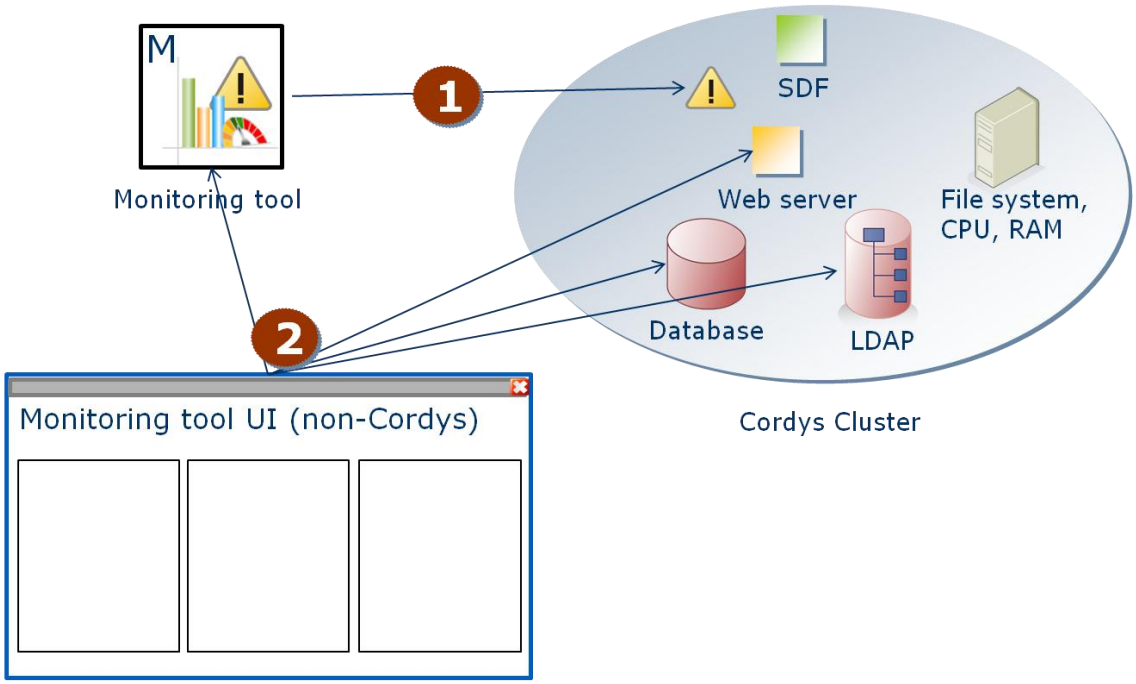


Figure 7.3.2 Data integration

The following steps are performed:

1. The monitoring tool detects a problem.
2. The UI displays data from the monitoring tool, and directly from Cordys components (e.g. using SOAP calls).

Pros

- Simple setup.
- Mostly independent.
- If Cordys is not operational, there is still much information available (log files, databases etc).

Cons

- Limited information available in the monitoring UI.
- If Cordys is not operational, only information from non-Cordys components is available (No SOAP calls).
- Using Cordys components, such as the engine tasks monitor, is only possible if a custom UI is built.
- All necessary Cordys logic must be build into the monitoring tool.

7.3.3 Scenario 3: UI integration (this scenario is deprecated, discontinued)

> This scenario is deprecated and inaccurate because the monitoring tool has its own user interface which is always available as a fallback (see next scenario).

In this scenario, the monitoring UI is created using X-Forms. The UI is a mash up integrating different Cordys components.

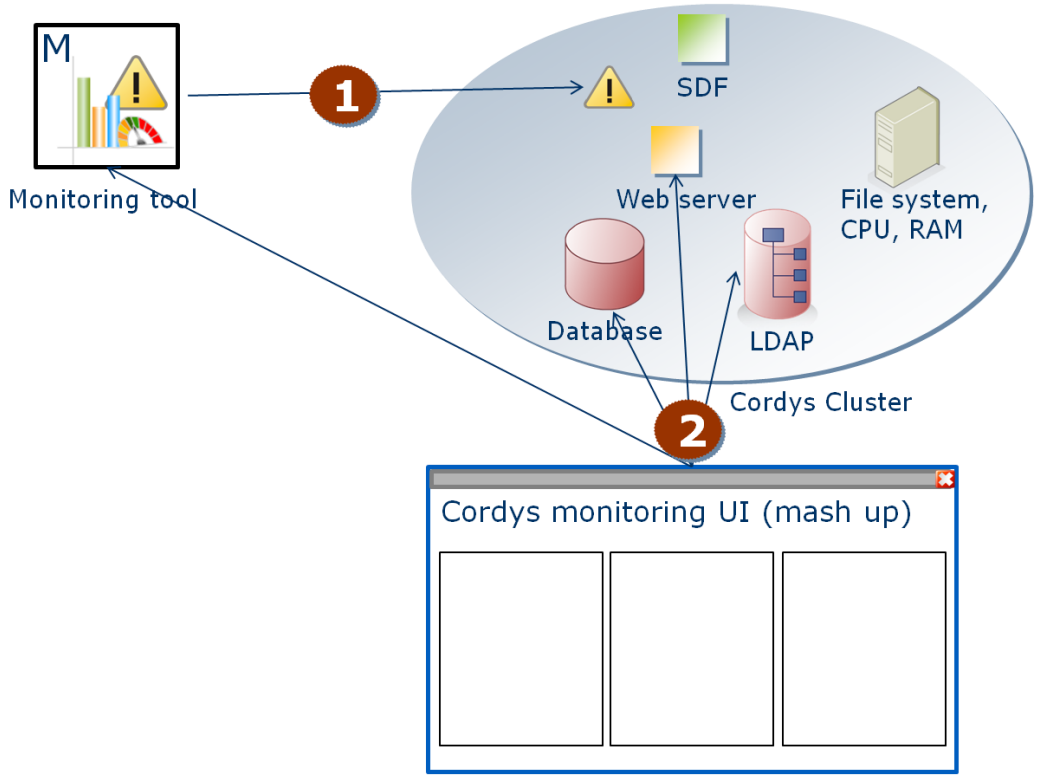


Figure 7.3.3 UI integration

The following steps are performed:

1. The monitoring tool detects a problem.
2. The Cordys X-Forms UI displays data from the monitoring tool, using SOAP calls, and directly from Cordys components.

Pros

- UI can be built using standard X-Forms.
- Standard Cordys components, such as the engine tasks monitor, can be easily used (no need to recreate Cordys functionality).

Cons

- If Cordys is not operational, the UI is not functioning, so there is no information available.

7.3.4 Scenario 4: Integration with fallback

The monitoring tool and the Cordys multi-cluster environment will be integrated. This will allow the use of Cordys features in workflow handling. The main UI is built in Cordys, combining several Cordys components and monitoring tool UI's into a mash up.

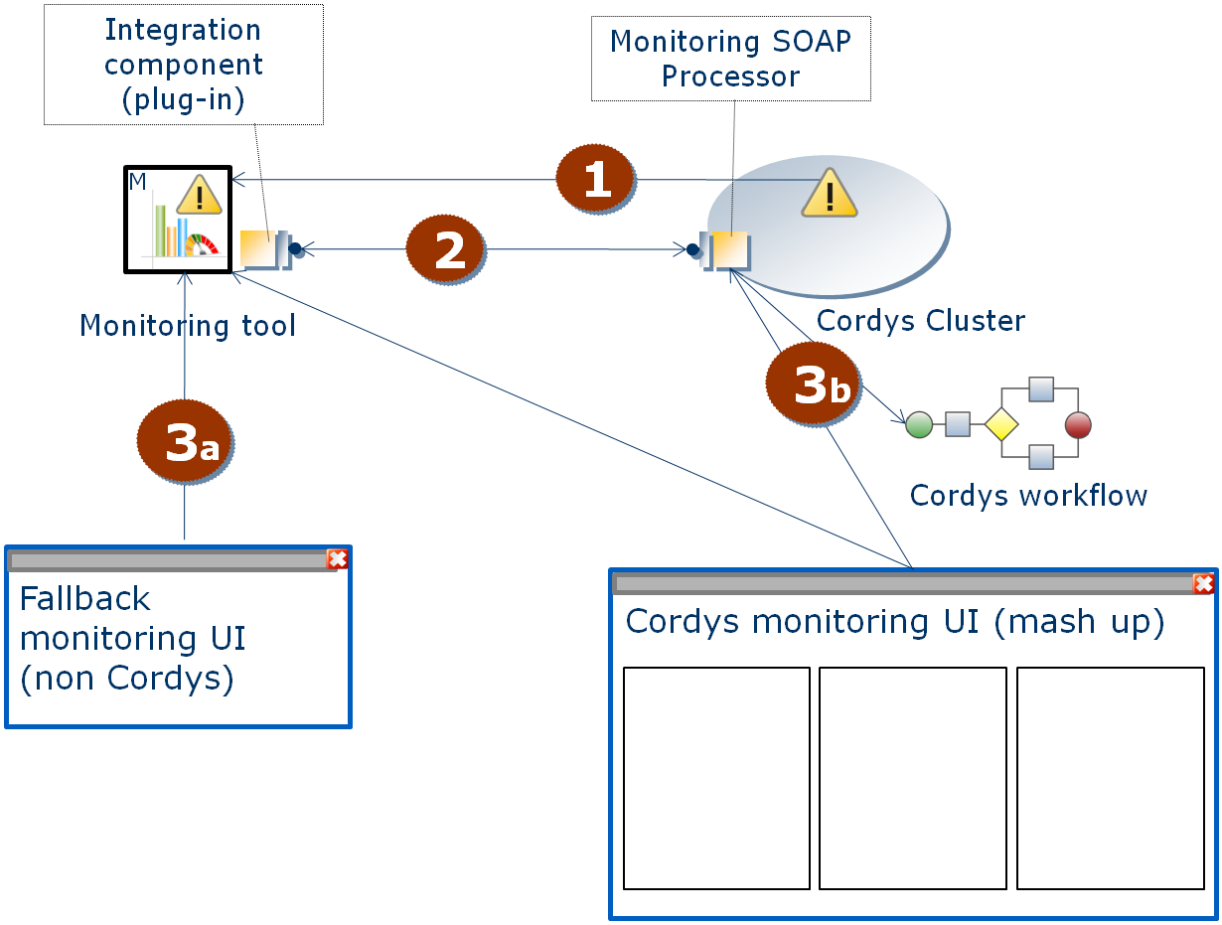


Figure 7.3.4 Integration steps

The following steps are performed:

1. The monitoring tool detects a problem.
2. The monitoring tool checks if Cordys is operational
3. Dependent of the result:
    a. Cordys is NOT operational:
    The monitoring tool autonomously handles the error using its own workflow capabilities.
    b. Cordys is operational:
    The monitoring tool transfers error handling to Cordys. E.g. using a SOAP call.

Pros

- Failover setup: if Cordys is not operational, the monitoring tool UI can be used for problem analysis
- All Cordys UI components can be used in created a monitoring UI.

Cons

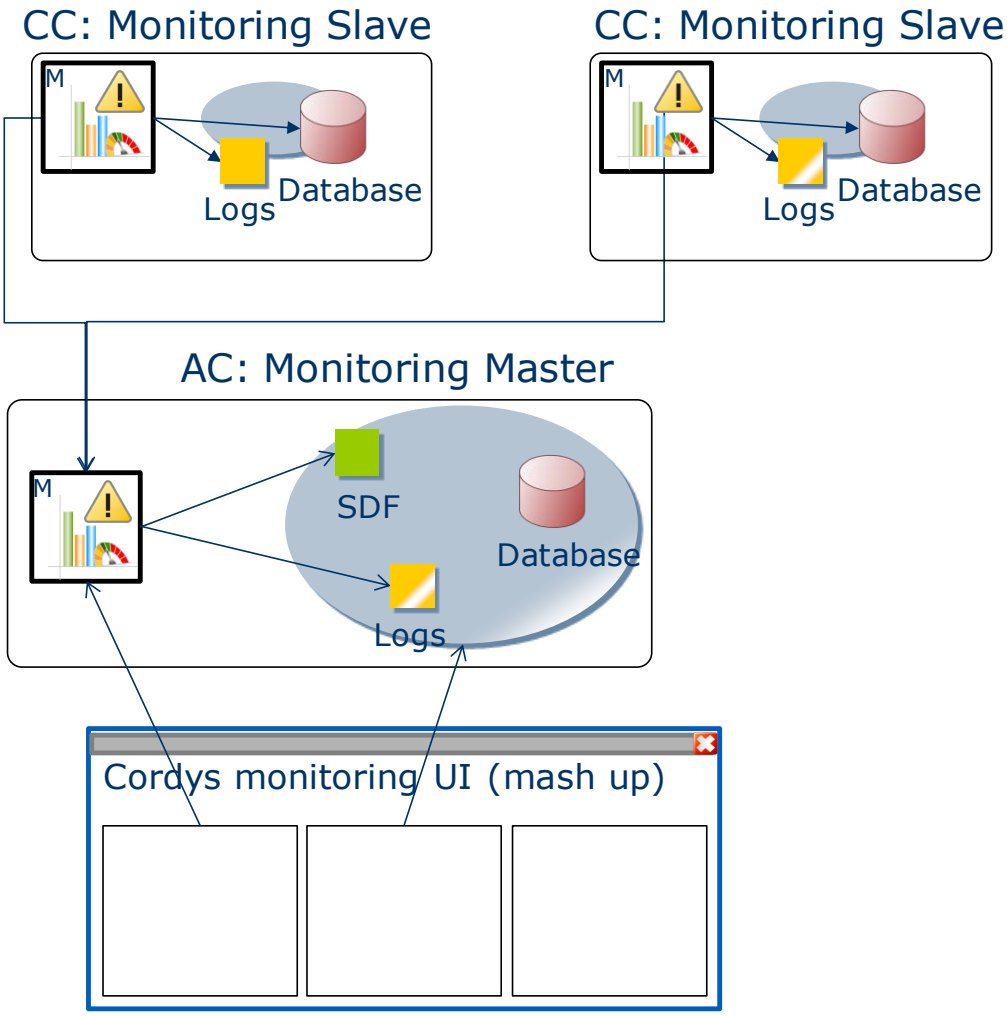
- Two UI's
- If Cordys is not operational, only a limited UI is available

7.4 Communication between monitoring instances

7.4.1 Keep the master monitor up-to-date.

Monitors slaves must communicate updates from their cluster to the master.

- The communication between monitoring instances must always be secure, or the data must be volatile.
- Monitoring instances must have a local data store to records events etc.
  This allows them to keep track of errors in case there is no connectivity with the monitoring master.
- The monitoring master must have the most up-to-date information.
  How this is achieved depends on the selected monitoring tool. Example methods could be: database synchronization, passing through events.



What must happen in case a link between clusters goes down or communication between monitoring instances fails?

- The administrator must be notified.
- This link is probably the same link the clusters are using to connect to the internet. Therefore sending alerts must be able to use different methods (e.g. texting gateway, dial-up).

7.4.2 Communication protocol

The communication between monitoring instances could happen using a proprietary protocol provided by the monitoring tool, or by using an interface which uses a standardized protocol.

Figure 7.4.1 Monitoring slave sends updates to the monitoring master.

When using a proprietary protocol:

- – Clusters must use the same monitoring tool.
- \+ Communication has less overhead.
- \+ No need to develop a communication interface for the monitoring tool.

When using an open standard protocol:

- \+ Different monitoring tools can be used on each cluster.
- – A monitoring tool which supports this standard is required, or a plug-in must be created.
- – Creating extra layers between monitoring instances increases the chance of failure.
- – This will create extra complexity.

Note: Which method is used to communicate between monitoring instances does not affect Cordys. This is only important when different monitoring tools are used at different sites.

8. Monitoring the Cordys Components

8.1 Introduction – Point of view

This chapter describes how the components of the Cordys Multi-cluster architecture will be monitored. The procedures are described from the viewpoint of a monitoring instance, located at the cluster site. Each monitoring instance checks these components for its own cluster.

> Note: Some components are only available at the Admin Cluster. These items are marked with " (AC only)"

8.2 Provisioning Tasks (AC only)

As the focus of this project is on cluster provisioning, this is the first component which must be monitored. This can be achieved using two methods.

8.2.1 Request tasks in error state using a SOAP call

This method uses a SOAP call to retrieve the list of provisioning tasks.

1. The monitor sends a SOAP request to the provisioning processor.
2. The SOAP processor receives the request.
3. The SOAP processor sends the list of the provisioning tasks in error state.
4. The monitor receives the SOAP message, and issues an alert if any tasks in error state were found.

Requirements

- The provisioning processor must be running.
- The monitor must be able to use SOAP.
- All components required to communicate and to handle SOAP calls must function properly.

8.2.2 Watch the provisioning database

This will be done by periodically checking the provisioning database.

1. The monitor opens a connection to the provisioning database.
2. A query to fetch the tasks with status error is send.
3. The database server returns a list of tasks in error status.
4. If there are any tasks in error, an alert is issued.

Requirements

- The provisioning database is running.
- The monitor must be able to communicate to a database directly (SQL plug-in).

8.3 SOAP Processors

As this is the most important part of Cordys, it is important that these will be monitored.

SOAP processors must be monitored using soap calls. This will allow the monitor to determine whether the SOAP processor is functioning properly.

1. The monitor fires a SOAP call to the SOAP processor.
2. The request arrives at the SOAP processor.
3. The SOAP processor handles the request, and sends a response.
4. If the monitor receives a correct SOAP response, the SOAP processor is running.

Requirements:

- The monitor must be able to use SOAP.
- All components required to communicate and to handle SOAP calls must function properly.

8.4 Log files

All Cordys components use log files to store errors. Log files are scattered across all machines in a cluster.

Log files can be watched for recent changes, and for keywords like ‘error’, ‘fail’, etc.

1. The monitor periodically checks the timestamps of log files.
2. If timestamps have changed recently, it reads the file.
3. It searches for given patterns and keywords (‘error’, ‘fail’, etc).
4. If it finds any occurrence, it issues an alert.

Requirements:

- The monitor knows which machines are in each cluster.
- The monitor knows the locations of all log files.
- The monitor has access to the log files on all machines.
- The monitor must be able to locate new information in files.

8.4.1 Log file discovery

As the requirements show, a method of discovering which log files must be monitored is necessary. Cordys keeps track of its log files in the LDAP database.

- The monitoring tool must periodically send a request to the local cluster to retrieve a list of log files per cluster-machine.
- Using this list, the monitoring tool knows which log files exist, and can start watching each of them.

8.4.2 Manual selection of log files

Selecting log files manually is only possible if the following conditions are met:

- The number of log files is fixed.
- The location of log files doesn’t change over time.
- The names of log files are fixed.

8.5 Web Gateway (HTTP Protocol)

Since all external Cordys communication use the web gateway, it is important to know the status of the web gateway (Apache). This can be achieved using a HTTP GET request.

1. The monitor fires a GET request using the HTTP protocol.
2. The request arrives at the web gateway.
3. The web gateway handles the request, and sends a response.
4. If the monitor receives a HTTP response, the web gateway is running, and the HTTP protocol is functioning properly.

Requirements:

- The web gateway is configured to respond at a simple HTTP GET request.
- The HTTP protocol is not blocked by any intermediate component.

8.6 Network connectivity & Response

Network connectivity must also be tested. This allows administrators to distinguish network problems from Cordys problems. This could also help in predicting errors. If the response times are increasing, possible time-outs can be expected, leading to higher level problems.

1. The monitor sends a ping request to each individual cluster.
2. Each cluster responds with an echo reply.
3. The monitor receives the reply and records the response time.
4. If no reply is received, an alert is issued.

Requirements

- Clusters must be configured to respond to a ping request.
- Intermediate components must allow ping traffic (firewalls).

> Note: Using the ping protocol is preferred in this situation because it detects the lowest level internet connectivity. If this protocol is not available, using another low level protocol is recommended (e.g. TCP or UDP port check).

8.7 SQL Database & LDAP database

Cordys uses a number of Databases to store its information; therefore the database server(s) must be monitored.

8.7.1 Execute test Query

This method tests if the database server is functioning properly.

1. The monitor sends a test query to the database server.
2. If the database server returns a normal result, the database server is fully operational.

Requirements

- Database server must be running and reachable.

8.7.2 Check server connectivity

This method only checks if the database server is reachable, but has less performance impact on the database server.

1. The monitor opens a TCP connection to the database server port.
2. If the port is open, the database server is reachable and running.

8.7.3 Check process

1. The monitor logs in onto the database server OS
2. The monitor checks if the database server process is running

Requirements

- The monitor has console access to the database server.

8.8 Mail server

Cordys uses a mail server to send users email. This could be monitored periodically.

1. The monitor opens an SMTP connection to the mail server using the TCP protocol.
2. The SMTP server sends a hello message.
3. If the monitor receives the message, the mail server is operational.

Requirements

- Intermediate components must allow SMTP.

If this mail server is the same server, the monitor is using to send email alerts; an alternate method must be available. Otherwise sending alerts would not be possible in case this mail server goes down.