jamk.fi

Transfer of monitoring solution

Adam Hrín

Bachelor's thesis
December 2019
School of Technology, Communication and Transport
Information and Communications Technology

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Description

Author(s)	Type of publication	Date
Hrín, Adam	Bachelor's thesis	December 2019
		Language of publication: English
	Number of pages: 54	Permission for web publication: yes

Title of publication
Transfer of monitoring solution

Degree programme
Information and Communications Technology

Supervisor(s)
Salmikangas, Esa

Assigned by
Solteq Oyj

Abstract

The customer of the assigner Solteq Oyj had requested for a transfer of a monitoring solution to a new platform. The currently used monitoring platform is an open source monitoring software Zabbix version 2.4. It performs primarily network, process and application monitoring of an instance of IBM WebSphere Commerce implementation.

Firstly, it was requested to review the existing monitoring and improve its current state. For that, it was required to understand how the monitoring platform has been configured. It was decided that visualization of the monitoring is to be done in a form of network maps. These maps could help the team to recognize problems, bottlenecks and downtimes even faster and more effectively as before.

The improvement of the current monitoring solution required configuration and programming in Bash scripting language.

During the process of visualizing the monitoring, new programming needs arose. New monitoring loopholes were found, and the purpose was to fix them. As a side product of this process, the monitoring documentation was improved and enhanced to an extent that the whole configuration was documented in one place.

The documentation also served as a specification for the transfer of the solution to a proprietary SolarWinds platform. A sample monitoring was ordered from the SolarWinds team in order to test the functionalities. The platform was studied and understood to the needed extent. The transfer process has been in progress and will continue in near future.

Keywords/tags (subjects)
Network monitoring, application monitoring, Zabbix, SolarWinds

Miscellaneous

Contents

Figures

Tables

1 Introduction

In today's world, the terms network and application monitoring are crucial in the IT industry. Network monitoring describes an IT process where networking components are monitored for fault and performance and maintaned continuously to optimize their availability. An important aspect of network monitoring is that it should be proactive. Proactivity helps to find issues and bottlenecks in the initial stage. Efficient proactive monitoring can prevent network downtime or failures. (Basics of network monitoring n.d.)

Application monitoring is a process that ensures that a software application processes and performs in an expected manner. Typically, application monitoring provides runtime metrics of system performance. These metrics include the transaction time, system response, transaction volume and overall health of the back-end infrastructure. Generally, the metrics are delivered in the form of graphical figures and statistics. These figures make it possible to evaluate the performance of an application or the overall application infrastructure. (Application monitoring n.d.)

There are several monitoring platforms on the market. The hosting company Solteq Oyj currently uses an open source platform. The customer of the company's team requested the monitoring to be transferred to a proprietary platform which the customer site already has in use.

The current monitoring solution serves the AMS (Application Maintanance Support) team to identify application and network issues. The monitoring platform offers an alerting functionality. In case when monitored process turns into problem state, this is alerted in an alerting dashboard and by an email notification. Naturally, the monitoring is not perfect and the network and application monitoring have several loopholes. It is required to imrove the current state of monitoring. During this process, visualizing the network in a form of network maps was requested.

After implementing the improvements to the current monitoring platform the transfer to another monitoring platform was realized.

The customer of the assigner required not to be mentioned in this thesis, therefore all references to the customer are anonymized.

2 Theoretical background

2.1 Importance of monitoring

In order to understand what needs to be improved during the implementation phase of the thesis, the most important objective needs to be explained. Monitoring the health of a software product, sanity of processed data and the network in which the software is deployed is equally important as the development of the product itself. During the application lifecycle, many changes and customizations need to be introduced as the application is further developed and new customers come. If monitoring is constructed well, it can notify the service team of an issue that can be resolved even before the consequences potentially reach the customer.

2.2 Network monitoring

Network monitoring is the use of a system that constantly monitors a computer network for slow or failing components and that notifies the network administrator in case of outages or other trouble. Commonly measured metrics are response time, availability and uptime, although both consistency and reliability metrics are starting to gain popularity. (Network monitoring n.d.)

Status request failures, e.g. when a connection cannot be established, it times-out, or the document or message cannot be retrieved, usually produce an action from the monitoring system. These actions vary; for example, an alarm may be sent (via SMS, email, etc.) to the resident sysadmin, automatic failover systems may be activated to remove the troubled server from duty until it can be repaired. (ibid.)

According to Olups (2019), basic functionality that can be expected from a monitoring solution is:

- **Data gathering**: This is where everything starts. Usually, data is gathered using various methods, including Simple Network Management Protocol (SNMP), agents, and Intelligent Platform Management Interface (IPMI).

- **Data storage**: Once the data has been gathered, it is beneficial to store it for later analysis.

- **Alerting**: Gathered data can be compared to thresholds and alerts sent out when required using different channels, such as email or SMS.

- **Visualization:** As the data has already been gathered and stored, it is trivial to generate simple graphs or maps from it. (Olups 2019, 2.)

According to Basics of network monitoring (n.d.), the most important aspects of network monitoring are:

- Monitoring the essentials: Identifying the devices that need to be monitored

- Optimizing the monitoring interval: Deciding on the frequency of polling

- Selecting the right protocol: Secure and low-bandwidth consuming network management protocol (e.g. SNMP)

- Setting thresholds: Aiming to identify real-time performance bottlenecks proactively by being alerted about an issue sufficiently long time before the network downtime

2.3 Agent-based and agentless monitoring

A decision about what kind of monitoring is going to be used needs to be made. It depends on what metrics need to be monitored and which protocols can be used. It is usually not an either-or decision, since some parts of the network can be monitored without agents while other parts require agents to be installed.

Agents are small software applications that are tailored to run on each device that the user wishes to monitor, collect desired data such as a performance metrics and

dependencies, and report that data back to a central repository - such as a monitoring server - which organizes, processes and visualizes that data. Agents are extremely common for monitoring heterogeneous environments, and they can be tailored to almost any system. Agents provide versatility; however, they also present additional workloads that IT administrators must manage and maintain. (Bigelow 2018.)

On the contrary, agentless products do not embed special management features into the hardware device. Instead, they rely on industry-standard interfaces to gather monitoring data. With standards-based communication interfaces, agentless systems provide lightweight monitoring that targets key metrics and basic monitoring situations. (Korzeniowski 2016.)

For agentless monitoring, implementation ranges from built-in SNMP agents to remote shell access, such as SSH. “Agentless” is slightly a misnomer. All management requires an agent, whether the agent is embedded in the management platform, the managed device, or a separately installed piece of software. The industry has accepted the de-facto definition of agentless as a management agent embedded in the software of the device or as a capability of the manager, requiring no separate installation or licensing. Agentless monitoring really means the use of existing, embedded capabilities. (Agent vs. Agentless Monitoring 2016.)

Agents provide more in-depth analysis and higher levels of security than agentless monitoring achieves but also can be harder to deploy and more expensive. (Korzeniowski 2016.)

2.4 Zabbix

There are multiple monitoring tools on the market, some of them being proprietary, which means that a license needs to be purchased in order to use the product (such as SolarWinds, PRTG, ManageEngine). The second option are open source platforms, such as Nagios and Zabbix. (Wilson 2019) The tool used for monitoring in the author’s team is currently Zabbix version 2.4.

Zabbix is an open source monitoring software tool for diverse IT components, including networks, servers, virtual machines (VMs) and cloud services. Zabbix provides

monitoring metrics, such as network utilization, CPU load and disk space consumption. The software monitors operations on Linux, Hewlett Packard Unix (HP-UX), Mac OS X, Solaris and other operating systems. (Rouse 2018.)

Zabbix can be deployed for agent-based and agentless monitoring. Agents are installed on IT components to check performance and collect data. The agent then reports back to a centralized Zabbix management server. That information is included in reports or presented visually in the Zabbix graphical user interface (GUI). If there are any issues regarding what is being monitored, Zabbix will send a notification or an alert to the user. Agentless monitoring accomplishes the same type of monitoring by using existing resources in a system or device to emulate an agent. (Rouse 2018.)

Architecture

Zabbix provides ways of monitoring different aspects of IT infrastructure. While many installations have a single central system, it is possible to use distributed monitoring with proxies and most installations will use Zabbix agents. (Olups 2019, 5.) A simple Zabbix setup depicting several monitoring capabilities can be seen in Figure 1.

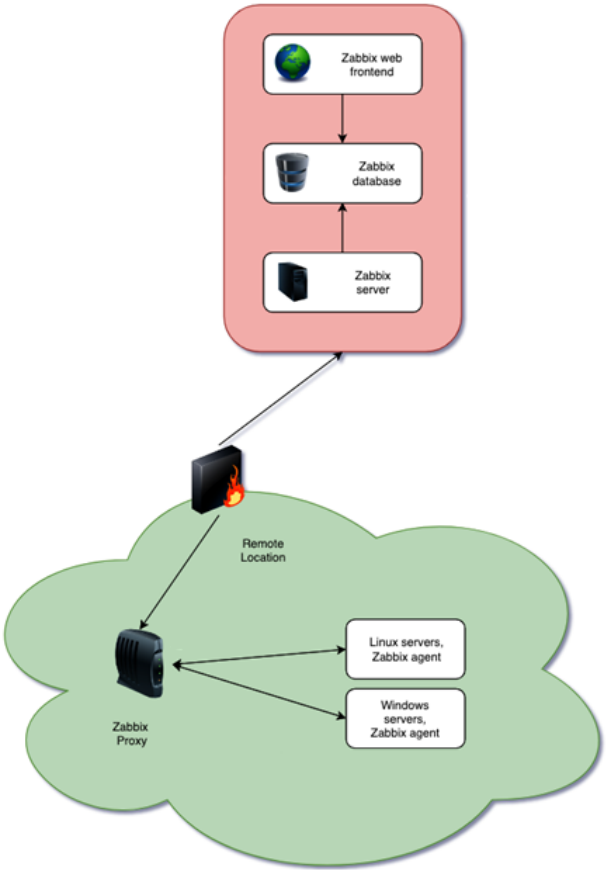


Figure 1. Example Zabbix setup (Olups 2019, 5.)

The Zabbix server directly monitors multiple devices but a remote location can be separated by a firewall, so it is easier to gather data through a Zabbix proxy. The Zabbix proxy and Zabbix agents, just like the server, are written in C language.

The main component is Zabbix database. Both Zabbix server and Zabbix frontend need an access to Zabbix database. Zabbix frontend needs an access to Zabbix server as well in order to display metrics about the server and some additional functionality. The required connection directions are shown in Figure 2. (Olups 2019, 7)

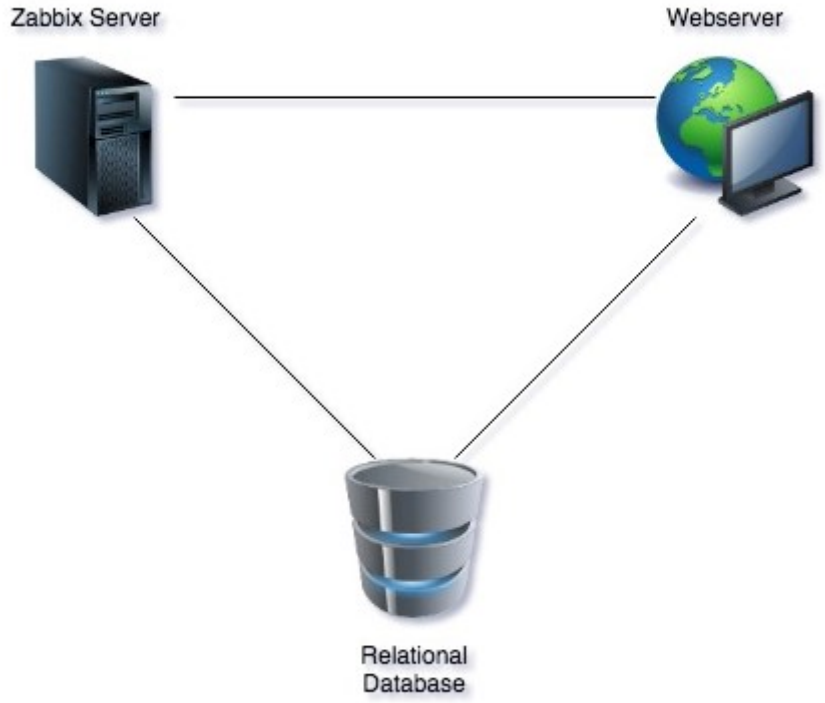


Figure 2. Zabbix architecture (Olups 2019, 7)

Zabbix agent

In Zabbix, the agent is a piece of software deployed on a monitoring target to actively monitor local resources and applications (hard drives, memory, processor statistics etc). The agent gathers operational information locally and reports data to Zabbix server for further processing using a JSON based communication protocol. In case of failures (such as a hard disk running full or a crashed service process), Zabbix server

can actively alert the administrators of the particular machine that reported the failure. Zabbix agents are extremely efficient because of use of native system calls for gathering statistical information. (Zabbix agent 2018.)

There are two types of checks Zabbix agent can perform: passive and active. In a passive check, the Zabbix server makes a data request towards agent and the Zabbix agent provides the result. In an active check, the agent must first download from the server a list of items to check. Then, based on the list, the agent will periodically send required values of specified items to the server. (ibid.)

Agentless monitoring in Zabbix

Besides of previously mentioned Zabbix agent options, there is another way how to perform monitoring of simple things without the need for an agent. Zabbix agentless monitoring is done with a help of Simple checks, which rely on basic network protocols such as ICMP and TCP to query monitored hosts (Olups 2019, 125). These checks are mainly used to check the basic reachability of a host, such as if TCP port is open or ICMP ping from the Zabbix server towards monitored host works.

Zabbix proxy

Zabbix proxy is a process that may collect monitoring data from one or more monitored devices and send the information to the Zabbix server, essentially working on behalf of the server. All collected data is buffered locally and then transferred to the Zabbix server the proxy belongs to. Deploying a proxy is optional, however, it may be very beneficial to distribute the load of a single Zabbix server. If only proxies collect data, processing on the server becomes less CPU and disk I/O hungry. (Zabbix proxy 2018.)

Zabbix features

In order to understand how Zabbix works the basic features need to be introduced. In Zabbix, *hosts* are the devices which are being monitored (servers, workstations, switches, etc). Hosts are organized into *host groups*, which group hosts logically based on various rules (platform-level, cluster-level rules, etc). Each host is configured with number of *items*. Items are the most essential individual metrics. It is specified what sort of data will be gathered from the host in an item. Items can be

logically grouped into *applications*. (Zabbix configuration 2018.) Olups (2019, 189) points out some of the most used item types to be Zabbix agents, simple checks, SNMP or IPMI.

Triggers

Triggers are logical expressions that "evaluate" data gathered by items and represent the current system state. Trigger expressions allow to define a threshold of what state of data is acceptable. Therefore, should the incoming data surpass the acceptable state, a trigger is "fired" or it changes the status to PROBLEM.

Whenever a trigger changes its status, Zabbix *event* is generated. This is essential for event correlation, which allows to correlate problem events to their resolution in a manner that is very precise and flexible. Other types of events are discovery events generated when hosts or services are detected, auto registration events generated when active agents are auto-registered by server, and internal events generated when an item or low-level discovery rule becomes unsupported or a trigger goes into an unknown state. (Zabbix configuration 2018.)

Each trigger is also defined by its severity. There are Information, Warning, Average High and Disaster severity options. The lowest generally describes the beginning and the end of normal business operation, the highest indicates some severe problem with immediate effect on the business. The actual severity meaning is defined by the user himself. Triggers of different severities are displayed with different colours on the user interface which brings a great visual help.

Discovery features

The previously mentioned discovery events are occuring as a result of discovery features. Discovery features can simplify administration and speed up Zabbix deployment. Network discovery feature enables Zabbix to automatically discover new nodes on the network. It is based on information such as result of periodical scans of IP ranges defined in Zabbix network discovery rules, availability of external services and information received from Zabbix or SNMP agent. The actions made upon the discovery result include creation or removal of hosts, adding hosts to host groups. Another type of discovery is the so called low-level discovery. This feature

provides a way to automatically create items, triggers and graphs for different entities on a computer. For instance, Zabbix can automatically start monitoring file systems, memory usage, CPU usage, disk space or network interfaces of hosts. (Zabbix discovery 2018.)

To ease up the process of assigning entities conveniently to multiple hosts, *templates* are introduced. The entities may be e.g. items, triggers, graphs, screens, applications, low-level discovery rules. Many hosts in real production can be identical or are fairly similar so naturally the sets of entities created for one host may be useful for many, which makes Zabbix configuration scalable and easy to manage. (Zabbix configuration 2018.)

User parameters

The items that allow to query the built-in capabilities of a Zabbix agent, query SNMP devices and reuse data on the Zabbix server are not always enough for some monitoring needs. Sometimes there is a requirement to monitor something that is not supported by Zabbix "out of the box". The easiest and most popular method to extend Zabbix data collection is user parameters. They are commands that are run by the Zabbix agent and the result is returned as an item value. (Olups 2019, 419.)

User parameters are configured on the agent side. The file of an agent daemon process `zabbix_agentd.conf` contains the definitions as follows:

```
UserParameter=<key>,<shell command>
```

This brings an enormous advantage of executing eventually any command on the Zabbix agent. According to Olups (2019, 421), it is recommended to use User parameters as active Zabbix items, as they can tie up server connections if they do not return a value very quickly.

Flexible parameters

To utilize the shell possibilities of using parameters in the command to make the item scalable, Zabbix offers Flexible user parameters, the syntax of which is as follows (Olups 2019, 422)

```
UserParameter=<key>[*],<shell command using $# shell parameters>
```

In this case, it is possible to specify parameters in the square brackets next to the key delimited by commas, and access them in the command using `$1, $2` shell syntax.

Wrapper scripts

Commands to be executed can be specified in the Zabbix agent daemon configuration file on a single line only. Pushing whole scripts there can be very messy and sometimes it can be hard to figure out the quotation. In such cases, a wrapper script has to be written. Such a script can be useful if parsing data requires more complex actions, or if parsing out multiple different values cannot be easily done with flexible user parameters. (Olups 2019, 435)

This solution is widely used in the author's team monitoring infrastructure, as it is easily scalable.

Zabbix visualization

It is fairly natural for humans to understand better visualized data. Visualization in Zabbix can be carried out with graphs which allow to grasp the data flow at a glance, correlate problems, discover when something started or make a presentation of when something might turn into a problem. The options for Zabbix graphs vary from simple one-item graph, more complex customized graphs to several-items comparison ad-hoc graphs. (Zabbix visualization 2018.)

Another visualizing option is network map. It is useful for having an overview of network's infrastucture depicting raised triggers on specific hosts or host groups. Connections between nodes can be customized to show occured problems between hosts using links. Application level monitoring can also be created utilizing network maps. (ibid.)

Very often there is a need to display more graphs or network maps on one page. Zabbix screens provide options for performing this feature. Essentially a screen is a table with any number of columns and rows, and each cell can display one element. These elements can be graphs, maps, other screens, entity overview information (entity being host, item, trigger, host group, etc.), problems by severity, clock, etc. (ibid.) Screens are significantly useful for network administrators, especially when

displayed on large screens, to have immediate information when something goes wrong somewhere on the network.

2.5 SolarWinds

The introduced Zabbix monitoring solution is currently used in the author's team. Due to a company decision, the servers running Zabbix server and Zabbix proxy will be shut down in near future; hence, the monitoring solution needs to be transferred. It has been requested by the customer that the platform used will be SolarWinds Orion platform.

In contrary with Zabbix, SolarWinds is a proprietary software for businesses to help manage their networks, systems, and information technology infrastructure. (Solar-Winds n.d.)

SolarWinds Orion Platform is a comprehensive bandwidth performance management and fault management application that allows you to view the real-time statistics of your network directly from your web browser. The Orion Network Performance Monitor will monitor and collect data from routers, switches, servers, and any other SNMP enabled device. Additionally, Orion platform monitors CPU Load, Memory utilization, and available Disk Space. (SolarWinds Orion Platform Integration n.d.)

Architecture

From the high-level point of view, the architecture of Orion solution is similar to the Zabbix architecture. Figure 3 displays a very simplified architecture of Orion platform. It consists of SolarWinds Orion Server responsible for all the data gathering and processing. The data is stored in an SQL Server and monitoring is presented to the user in a SolarWinds Orion Web Console. The application server monitors devices which can be basic network devices such as switches and routers, servers or all apps and websites. (One platform to rule your IT stack n.d.)

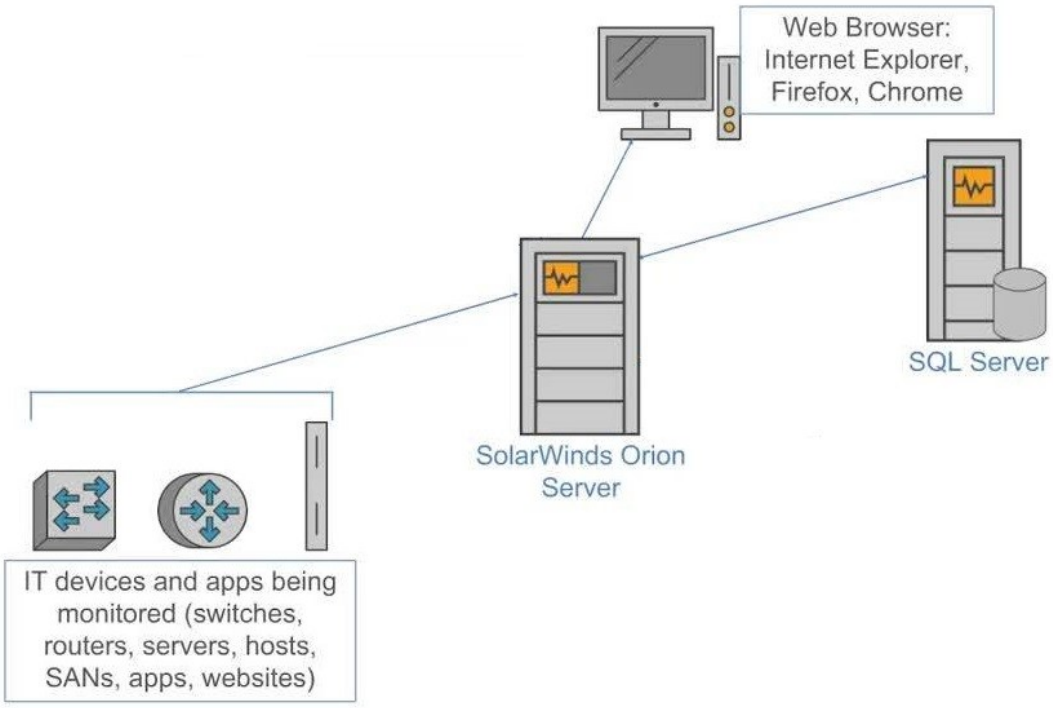


Figure 3. SolarWinds Orion platform architecture (One platform to rule your IT stack n.d.)

Features

SolarWinds comes with a wide variety of features and monitoring possibilities which are out of scope of this thesis. However, the focus will be on describing only the features which are going to be utilized when implementing the monitoring transfer.

The SolarWinds Orion web-console based tool which allows monitoring servers and applications is called Server and Application Monitor (SAM). SAM includes over 250 "out of the box" *application monitor templates* that can be assigned to *nodes* and used immediately. These templates are comprised of code and scripts that can be customized for individual nodes, or groups of nodes. *Alerts* and thresholds for monitored values can be configured. When polling occurs, scripts automatically gather data and report results within the Orion Web Console. (Server&Application Monitor n.d.)

Each template includes one or more *component* monitors designed to monitor a server, application, or process. These pre-built templates can be assigned to nodes to create *applications* that are specific to that node. (ibid.)

Figure 4 (Server&Application Monitor n.d.) illustrates how different templates can be assigned to nodes to create application monitors that display polling results in the Orion Web Console together with the web console frontend.

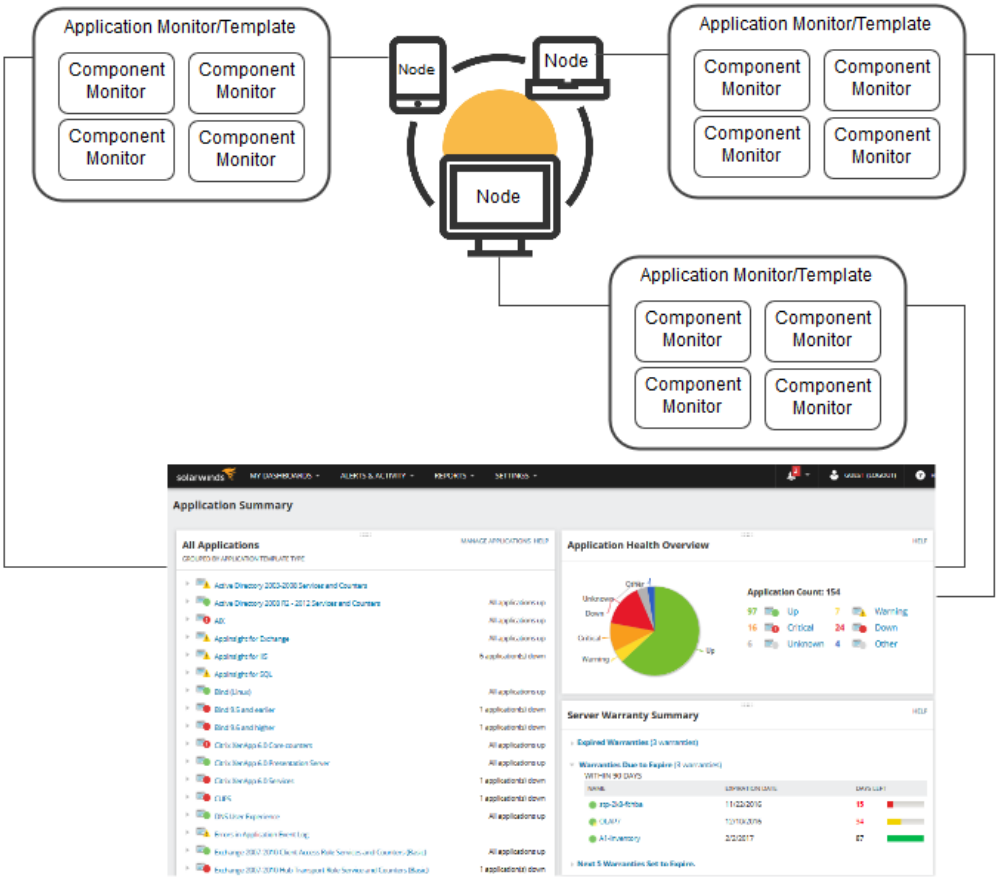


Figure 4. Server and Application Monitor (SAM) infrastructure (Server&Application Monitor n.d.)

2.6 Monitored system

The monitoring makes sense only if there is a system to monitor. The purpose of this chapter is to introduce the company, the system the author's team is working on and the most essential processes inside the system which are critical for the customer that hence are the most important to monitor.

Company background

The author of this thesis works as a trainee in a Finnish company Solteq Oyj in a team providing implementation and support of an e-commerce solution. The author's

work position is in an AMS department (Application Maintenance Support) which is responsible for maintenance and support of the product. This includes e.g. fixing bugs, updating APIs, modifying features, adding new features, enhancing app security, improving UI and UX, updating the app to keep up with platform and hardware requirements, and managing downtimes (Kitili 2018).

IBM WebSphere Commerce

The product that the team is delivering is IBM WebSphere Commerce. WebSphere Commerce is a single, unified e-commerce platform that offers the ability to do business directly with consumers (B2C), directly with businesses (B2B), and indirectly through channel partners (indirect business models). WebSphere Commerce is designed to be a customizable, scalable, and high availability solution that is built to leverage open standards. It provides easy-to-use tools for business users to centrally manage a cross-channel strategy. (WebSphere Commerce Version 8 n.d.)

E-commerce in general can be divided into several groups depending on who is the customer. If the customer is another business, term B2B e-commerce is used. On the other hand, if the customer is a single person considered to be the end customer, this is called B2C e-commerce. From the perspective of the WebSphere Commerce software, there are differences in implementation and service support in both stores. The majority of stores currently running in the author's team are B2B stores.

WebSphere Commerce Integration

External systems integration is a key feature of the WebSphere Commerce solution. In WebSphere Commerce business logic is enabled for integration and built-in adapters and interfaces are provided for common integration points. (Integrating n.d.)

WebSphere Commerce can be configured to send a message to a back-end system whenever an order is created at the store. This order information can be used by the back-end system to do necessary order fulfillment processing. The back-end system can later send order status messages back to WebSphere Commerce indicating that order delivery occurred, or an order invoice is issued. An email can also be sent to update the customer. (Integrating n.d.)

DataLoad Utility

It is important to understand processes running inside the WebSphere Commerce system. One of the most important processes is dataload process. All the data coming to the webshop must be understood by the system in order for the system to be able to store the data into a database. Data sources vary across the stores. Stores usually utilize their own ERP systems (Enterprise Planning System) as a data source for the webshop. IBM WebSphere Commerce DataLoad Utility is a prefered way how to load data into webshop. The loaded data are complex; however, from a higher perspective, four main data entities essential for any B2B e-commerce generally can be recognized: categories, products, pricelists and customers. It is important to realize that in B2B e-commerce, every customer can have different pricelist. They can obtain different promotions and discounts based on their previous purchases. Therefore, the pricelists, even for the same products, vary across the customers. According to Data Load utility architecture overview (n.d.), the simplified visual overview of how the dataload process works can be described according to IBM documentation as follows in Figure 5 (Data Load utility architecture overview n.d.):

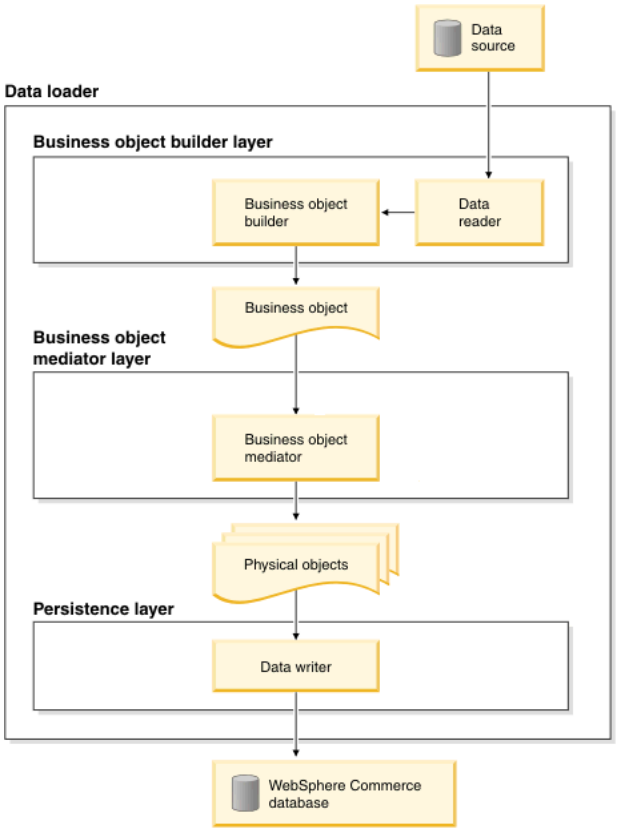


Figure 5. Dataload process (Data Load utility architecture overview n.d.)

1. In the Business object builder layer, the Data reader is responsible for passing the raw data source to Business object builder for processing and building business objects. The business objects are common entities throughout the WebSphere Commerce. The business object layer supports different types of data sources, such as CSV format, XML files, external databases or ERP systems.
2. The business object mediator converts the business objects into objects that represent the physical database schema – physical objects.
3. The persistence layer persists the physical objects into the WebSphere Commerce database.

Sterling Integrator

Every store may send the data in a slightly different format, which is why an integration system able to translate the data to the format understood by the DataLoad Utility must be placed in between.

Sterling B2B Integrator is a transaction engine that runs processes and manages them according to business requirements. One of the most important processes is data transformation. Data transformation is the cornerstone of electronic commerce. With Sterling B2B Integrator, data transformation can be manipulated easily. Supported data formats include Electronic Data Interchange (EDI), positional, variable-length-delimited, Japanese Center for Informatization of Industry (CII), and Extensible Markup Language (XML). (Sterling B2B Integrator n.d.)

3 Improving current monitoring

3.1 Current state

The author's primary task is to take care of the process of transferring monitoring solution from Zabbix to SolarWinds monitoring system. Before the transfer can be implemented, the current situation of Zabbix monitoring needs to be revised. The primary reason for this is that despite the fact that the team had created a unified procedure of creating templates, items and triggers, the procedure was not always

obeyed. This was causing that the Zabbix frontend started to be disarranged and inconsistent. The aim was to make the documentation of current monitoring complete and organized.

During the process of documenting and organizing a new challenge was expected to arise. An incomplete or wrong monitoring of part of the network can be revealed, eventually some parts can be found not to be monitored at all. The purpose of the work in this phase was to create or repair such monitoring with respect of how big advantage it brings to the team, thus to the customer.

Another reason for the current situation mapping was to help the author to adapt and get to know the system. The knowledge is expected to be passed to other team members in a form of an easy-to-access documentation.

3.2 Zabbix frontend

The Zabbix frontend can be accessed with different administration privileges. It is enough for a user who is interested only in the monitoring results to have an access to monitoring overview, historical data, network maps, screens, et cetera. Monitoring alert dashboard (shown in Figure 6) is the primary source of information about raised alerts.

Last 20 issues

Host	Issue	Last change	Age	Info	Ack	Actions
PROD WCS Staging	NO NEW SI ORDERCREATION files for 2 hour (by count)	2019-11-06 09:00:44	1m 55s		No	1 1
PROD SI 1	CPU usage > 90% (15 min avg)	2019-11-06 08:45:36	17m 3s		No	2 1
PROD SI 2	Sterling Integrator FTP - wrong response code	2019-11-06 08:36:45	25m 54s		No	1 1
PROD SI 2	Sterling Integrator FTP - response time 0ms?	2019-11-06 08:35:46	26m 53s		No	1 1
PROD SI 2	Sterling Integrator HTTP - wrong response code	2019-11-06 08:35:46	26m 53s		No	1 1
PROD SI 2	Sterling Integrator HTTP - response time 0ms?	2019-11-06 08:35:46	26m 53s		No	1 1
PROD SI 1	SI Adapter Containers are DOWN	2019-11-06 08:28:51	33m 48s		No	2 1
PROD SI 2	SI Adapter Containers are DOWN	2019-11-06 08:26:43	35m 56s		No	2 1
PROD WCS Staging	dataload.sh Trioving has been running for >60 minutes	2019-11-06 01:59:11	7h 3m 28s		No	1 1
PROD WCS Staging	dataload.sh Trioving running	2019-11-06 01:01:06	8h 1m 33s		No	1 1
QA WCS Staging	Stagingprop has exited with non-success code 203	2019-11-05 13:32:37	19h 30m 2s		No	
QA WCS Staging	10% swap used	2019-10-31 03:19:13	6d 5h 43m		No	2 1
TEST WCS Staging	10% swap used	2019-10-30 01:13:17	7d 7h 49m		No	1 1
PROD SI DB2 Primary	DB2 Reorg errors found for db2inst1/SI52DB	2019-10-27 09:07:19	9d 23h 55m		No	2 1
QA WCS Staging	NO NEW SI ORDERCREATION files for 2 hour (by count)	2019-07-26 22:13:25	3m 12d 11h	?	No	
QA WCS Staging	12 order(s) in OneShop V8 stuck in status M for more than 10 minutes!	2019-03-25 21:11:09	7m 15d 11h		No	
PROD WCS Staging	11 order(s) older than 24h have transferstatus 1 [PROD]	2018-03-26 19:02:40	1y 7m 14d		No events	

Figure 6. Zabbix dashboard

The dashboard contains information such as on which host the alert was raised, what is the issue, how long is the problem event fired and some additional information. For the sake of this thesis, the server DNS names were left out from the host column. The background colors of the alert texts represent the severities of the triggers. Based on this information, the team can perform all the necessary steps to fix the problem underlying the monitoring unit. The internal team's documentation should therefore be in place.

Additionally, the user with administrator privileges is able to configure parts of the monitoring. This user can then introduce new items, triggers and alerts as well as modify existing ones, configure network maps, templates, host groups and discovery events.

3.3 Documentation and configuration

Before the author began working on the monitoring solution, there had already existed a documentation on company's internal knowledge sharing system. This documentation begins with describing the team defined trigger severities meanings in detail. The main part of the documentation is a table describing the meaning and purpose of Zabbix items. The table shows which template the items are on, what triggers they have, which hosts are monitored by these items and, most importantly, what action should be taken when the trigger of certain item is active. An example of a record from the table can be found in Appendix 1.

There is already one inconsistency in the table, missing "Actions when triggered" column record for the last example. There are more similar cases in the documentation. For the team this means that if the trigger gets active, it takes longer time for the team member to find the cause. It is supposed that the creator of the Zabbix item might have had more insights about the problem and could have shared his knowledge to make the job easier in the future.

Definition of item keys

Another thing to notice in the table is the two kinds of item keys. All the Zabbix item keys in the team's configuration can be divided into two groups:

- Zabbix default keys
- User parameter keys

The Zabbix default keys are keys that come with the Zabbix “out of the box”. The Zabbix agent can interpret the keys with no additional scripting required. These keys usually include the most basic checks that are CPU, free disk space or available memory related. Other checks worth mentioning belonging to this category used in team’s monitoring solution are checking whether number of running processes on server has not exceeded a threshold, network traffic flow on server or system uptime of server. In this case, presented in Appendix 1, the example of Zabbix default key is `net.tcp.service[tcp,,8080]` which checks if Jenkins service is running. It is known that this service runs on given monitored host on port 8080. Using TCP protocol, Zabbix server checks the status of the service periodically.

It is important to mention that in the architecture of the Zabbix system there exists the Zabbix proxy that acts on behalf of the server and intends to lower the load off of the server. Therefore, the exact description of the previously mentioned check is that the Zabbix proxy checks the status of the service periodically and hands out this information to the Zabbix server. This applies for all Zabbix checks using agents.

The User parameter keys (in the example in Appendix 1 they are `checkDbBackupStatus` and `checkSIFTPHTTPCode)` make use of a custom shell script on the monitored host itself. In the configuration file of Zabbix agent located on the host there is an *Include* parameter. This parameter enables to include files in the configuration file as a wrapper scripts. When installing Zabbix, the default directory for these included files is generated as `<path>/zabbix_agentd.conf.d/*.conf`, meaning all the files in this directory with a suffix `.conf` will be included. There are three files with such a suffix: `zabbix_agentd.userparams.conf`, `solteq.conf`, and on database servers `db2.conf`. All of them consist of definitions of User parameter keys and their respective shell commands. The first file contains general checks such as CPU, System information or process check with only a one-liner shell command included directly in the file. An example of a check monitoring application processes is `findprocess` user parameter:

```
UserParameter=findprocess[*],[ -z "$3" ] && { ps -fU "$1" | grep "$2"
| wc -l; } || { ps -fU "$1" | grep "$2" | grep "$3" | wc -l; }
```

The `db2.conf` file contains database related checks such as High Availability Disaster Recovery (HADR) status, backup and logfiles monitoring.

The `solteq.conf` file contains all the specific checks per template or host. In the shell command section of each User parameter definition, a shell wrapper script `monitoring.sh` is referenced. This script contains all the functions performed by the agent on the hosts that return values to the Zabbix server. The definitions of keys from the example in Appendix 1 are:

```
UserParameter=checkDbBackupStatus<path>/zabbix_agentd.conf.d/monitor-
ing.sh checkDbBackupStatus
UserParameter=checkSIFTPHTTPCode,<path>/zabbix_agentd.conf.d/monitor-
ing.sh checkSIFTPHTTPCode
```

For the completeness, using flexible parameters with arguments passed to `monitoring.sh` functions is also possible in this fashion:

```
UserParameter=checkFtpFileCount[*],{ <path>/zabbix_agentd.conf.d/moni-
toring.sh checkFtpFileCount $1 $2; }
```

3.4 Visualization of monitoring in Zabbix maps

In the next phase of monitoring improvement a Zabbix maps feature is utilized. Zabbix maps visualize the system architecture with respect to connection between nodes. Each node and connection between nodes can be associated with items and triggers connected to that node or connection.

This process of creating maps has two main reasons. Firstly, it helps the team members to quickly spot new triggers raised and at the same time identify on which node the problem is. For the future team members, maps help to grasp the architecture of the system much easier and faster. Secondly, during the process of creating the map there can occur a need for new monitoring item to be made. Making the map is a creative process requiring logical thinking about nodes and their connections. Therefore, it is much more natural to reveal some missing monitoring item on a place where it should be.

The first and most obvious indicator is when there is no monitoring on the connection *between* nodes. This does not necessarily mean that the monitoring should be placed there because not all connections are needed to be monitored. However, it is a good place to start. Naturally, some nodes and connections need to be monitored with multiple items focusing on different aspects of the application or network. The maps help realizing what those aspects are.

Staging server

Before digging deeper into the monitoring, a specialized Staging server must be introduced to understand the underlying structure. The WebSphere Commerce Staging server is a part of the production environment where business and technical users can update and manage store data and preview changes. The changes can then be propagated to the production server. (Staging server n.d.)

Aside from its importance in the production application environment, Staging server is also important for monitoring. Most of the Zabbix checks regarding the specific features and application services are performed via Staging server. Some amount of checks is performed via other servers, an example of those are checks of connections or accessibility of services, availability of processes.

3.4.1 Visualizing the first process

Some customers use so-called Real-time pricing (RTP) services for providing net prices of their products to WebSphere Commerce (WCS). These services are external services called from WCS through Sterling Integrator (SI). SI works as a mediator that receives a SOAP request `GetPrices` from WCS over HTTP and passes this request in a form of service call to external RTP service. RTP service reacts to this call with a response back to SI and SI processes the response back to a format understandable by WCS. Finally, SI sends SOAP message `Prices` back to WCS over HTTP.

This process is critical for customers utilizing the RTP service, so it is important to stay notified of any problem on the route. There already exists an item `check-IsRealTimePricingUp[*]` in Zabbix. This item performs a check in `monitoring.sh` wrapper script sending a SOAP message to given URL and waits for the response. If the response takes too long to receive or it has non-success response code in HTTP

header the problem is signalized to Zabbix. Important part is the URL, which is specified based on the parameter given to the Zabbix item represented as the * symbol in the item definition. The item can check either the connection to RTP through SI sending a request to SI the same way WCS does, or it can check directly the external RTP service using its URL. For this monitoring check the Staging server is used.

Creating the network map

Visualizing this process into a map means taking the related servers as nodes and creating connections between them. One note regarding all the maps shown in this thesis is that the details of the nodes such as the DNS names of the servers, private IPs or other parameters have for the sake of the thesis been disregarded from the map since they are internal information. Figure 7 shows the RTP network map.

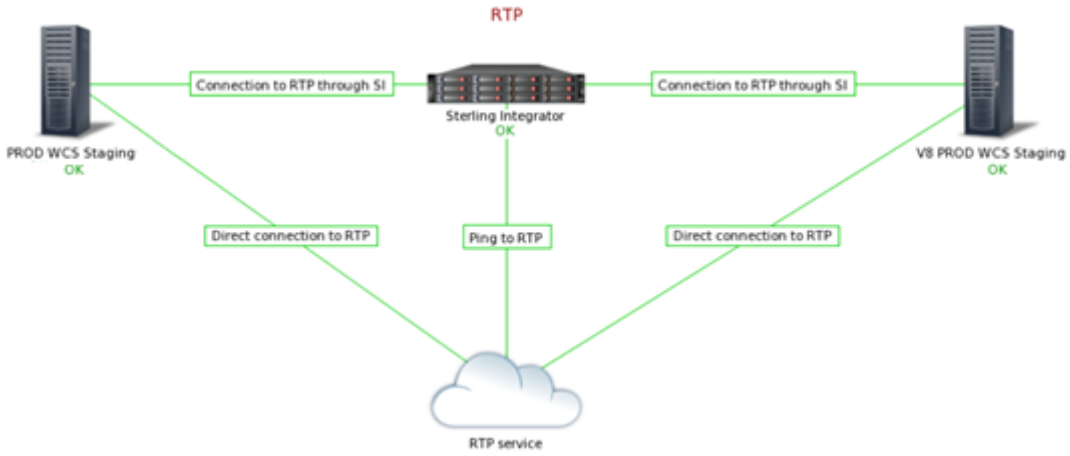


Figure 7. Zabbix network map of RTP service call flow

The map contains two Staging servers each representing the production Staging server in different WCS versions. Currently, both version 7 and version 8 of WCS are in production use, since some customers have their eshops in one version and some customers in the other. Additionaly there is a *Sterling Integrator* node and node representing the RTP services.

There is a certain level of abstraction on each of the maps. Firstly, in fact there are two RTP services represented in the production, which are on the map represented

by the cloud node. The line between this node and the Staging servers represents the connection to both the RTP services.

Secondly, the Sterling Integrator is a grouping of more servers hidden behind. The node is configured the way that it is possible to see one abstraction level lower – to see the architecture of Sterling Integrator itself. When the node is clicked, a menu with the option to see a *Submap* is visible under *URLs* submenu, as shown in Figure 8. After selecting the option user is presented with the SI network map, as presented in Figure 9.

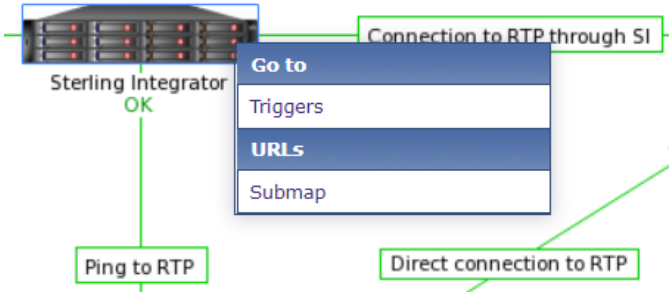


Figure 8. Network map node menu

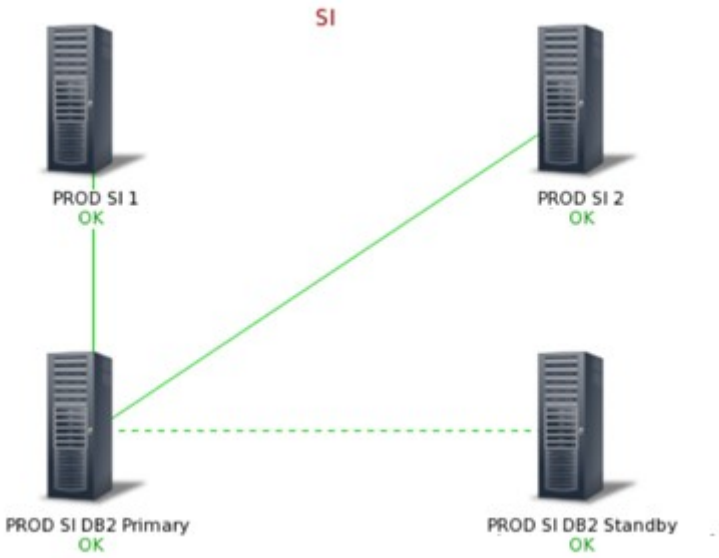


Figure 9. Sterling Integrator network map

The configuration of the nodes and connections between them is the core of the whole map. Each node represents one host or host group in Zabbix. There is a visual sign of when some trigger is active on a host, shown as a circle around the map node with a color attributable to the severity of the trigger. However, it is not desirable to show every active trigger of the server. For example, as mentioned before, Staging server is responsible for most of the monitoring items, thus has a lot of triggers that can be active. In this case, it is required to configure the nodes to signalize only active triggers related to RTP monitoring. This can be achieved with help of applications. Every item in Zabbix must belong to some application which groups items logically. This application can then be configured on the map node, which essentially means that only triggers assocatied with that application's items will influence the node. For the Staging servers this means that it is required to configure the nodes to accept only the `checkIsRealTimePricingUp` application. In this case there is only one item `checkIsRealTimePricingUp[*]` which belongs to this application since no other monitoring check is made from the Staging servers towards SI or RTP services.

The *Sterling Integrator* node is in fact configured as a host group node. A new host group *SI PROD* was created in Zabbix which groups six production SI hosts. Apart from the four SI servers visible on figure 9, there are two more servers – SI Perimeter 1 and SI Perimeter 2; however, for the scope of this monitoring they are not important.

For the host group node, there is also an option to define an application in the configuration. However, there are no applications on the host group level in Zabbix. If a trigger related to the application on any host inside the host group is active, the host group node will signalize it with a coloured circle around the node. Unfortunately, Zabbix host group map nodes do not show which host the active trigger belongs to directly on the map. To find it out, the user must click the node and choose a *Triggers* option in the menu under *Go to* submenu (figure 8), which will redirect him to active triggers list with respective hosts shown.

The connections between nodes represented by lines with labels on them can also be configured to change appearance when some trigger is active. For example, the line can go red or bold to emphasize problem, alternatively the line can turn into dashed or dotted line. The options how the line is shown if a problem state occurs are

completely customizable. The lines are configured separately by choosing related triggers. Hence, the line between Staging servers and SI will change to bold red when the RTP via SI request results in a problem. Similarly, when the direct request between Staging servers and RTP services fails, the respective lines turn bold red.

There is one additional line with a label *Ping to RTP* from SI to RTP. An existing item `checkPingIntervalRTP` is utilized. The item makes a simple ICMP ping to RTP services from SI nodes and evaluates the response. There are two triggers configured for this item, one firing when the ping times out and one when the ping takes in average longer time to response than some preconfigured treshold. Both these triggers are configured on the map, so whenever they are active, the map shows it.

By creating the map the main goal was achieved. During the process an unsatisfactory configuration of triggers had been realized and was needed to be improved. It was found out that existing triggers had not been configured properly and had used Zabbix trigger functions in a wrong way. It has been concluded that two kinds of triggers for each connection will be created. The connections are between Staging and SI, between Staging and the first RTP service and between Staging and the second RTP service. One trigger activates when the response from the respective server has not reached the sender and another trigger activates when there was no response for fifteen minutes. The trigger expressions look like this for check through SI:

```
{checkIsRealTimePricingUp[si].last()}=0
{checkIsRealTimePricingUp[si].count(900)}=0
```

The trigger function `last()` with no arguments gives the last measured value. The trigger function `count(900)` gives a sum of the values measured in last 900 seconds (15 minutes). The item is configured to run the check every 60 seconds.

In the case of the first trigger firing, the team members should be prepared to take an action, while when the second one fires they are supposed to take the action immediately.

3.4.2 Order creation process

The most important feature of e-commerce solution is making orders. The shopper can add product to a shopping cart creating an order by submitting the purchase. The order undergoes different processes and transfers through various states during its lifecycle. When the order is submitted in web shop frontend, a copy of the order is created in WebSphere Commerce. The first status for the order is *Submitted*. The order transfers to Sterling Integrator for processing in a form of ORDERCREATION xml file. The output of SI processing is a file in a format that the external ERP systems can receive. The file is stored in an Output folder on SI server. There is a periodical operation of pulling these files to FTP server, where they wait until an external ERP system picks them up. As a backup, every processed ORDERCREATION file is stored in Staging server in an archive. Once the ERP system confirms the order with an ORDERDETAIL xml file sent back to WCS, the status of the order is changed to *Confirmed.* Similarly, during processing of the order, update messages about shipment and invoicing are sent back to WCS. Based on the information returned by ERP, the order states can update to *Partly shipped* if only a part of ordered items are shipped to customer's shipping address or *Shipped* if the whole order is successfully shipped. Subsequently, order status can change to *Partly invoiced* if only a part of ordered items is invoiced or *Invoiced* if the whole order is invoiced.

Visualization of Order creation process

The concept of creating and configuring the Zabbix map is similar to the RTP process case. In this map only the Order creation process itself is monitored in the direction from WCS to ERP (from the left to the right on the map). The detail messages flow coming from the external ERPs to WCS are not monitored at this time. The map can be found as Appendix 2.

The database monitoring (the leftmost DB2 node) contains several checks. It is indicated in which status and transfer-status the order is in the database record. The transfer-status indicates whether the order has already been sent to Sterling Integrator (SI) for processing. It was observed that sometimes the order is already in transferring state while the order is not in *Submitted* status. One check is dedicated to alert about these orders. Another check is made to find out whether any

ORDERCREATION files have been created recently. If there is no new file in the last two hours, this indicates a problem and an alert is shown about it.

There was a need to create new monitoring items utilizing the database data. If an order is stuck in *Submitted* status for more than 10 minutes, this means that the order was never delivered to SI for processing and thus, it never goes further in the flow. Additionally, the order can get into 'H' status. which indicates that transfer to SI was attempted but there was an issue (i.e. SI was down). In this case, the order will be automatically retried to be transferred again by a scheduler running hourly. Therefore, if the order is still stuck in the 'H' status after two hours, this means a problem and an alert should be raised. Similarly, an order could end up in an 'F' status. This would indicate an unknown issue with SI and some manual actions need to be performed.

All these checks are implemented to be performed in similar ways. There is a Jenkins job running scripts on Staging servers in different intervals. Jenkins is an automation platform which helps to automate the non-human part of software development process and can execute arbitrary shell scripts (Jenkins (software) n.d.).

Every time the script runs, it retrieves the current stuck orders from the database and compares this list against the stuck orders found in previous script run. If there are any new stuck orders, the order numbers are sent via email with a message alerting existence of such orders. The number of stuck orders is returned as an output of the script so that Zabbix can show the alert containing the amount of these orders. A record about previously found stuck orders is kept in the logfile until the next run of the script.

Another node monitored is WCS APP – the WebSphere Commerce Application server. More specifically, the connection from WCS APP to SI is checked via SOAP interface. If there is any problem with the response from SI back to WCS APP an alert is shown. This problem is also indicated on the connection between WCS APP and SI Load Balancer nodes on the Zabbix map.

The SI Load Balancer node represents the Sterling Integrator. In this specific case from the Appendix 2 there are two problems indicated on the node, which means two triggers got raised. The blue circle around the node tells the severity of the alert.

Blue color is associated with *Information* severity (*Warning* severity would be shown in yellow color, *Average* severity is orange and *High* severity is represented by red color). In case of multiple alerts active at the same time the highest severity is indicated. Unfortunately, if there are two or more problems the map does not reveal the texts of the alerts. Instead, the node needs to be clicked on and *Triggers* option in the menu should be chosen in order for the list of active alerts to be shown.

Monitored functionalities on the SI Load Balancer include checking responsiveness of SI FTP internal adapter and checking SI SOAP interface availability recursively.

The SI FTP server is in fact part of the SI servers – it is internal FTP accessible via adapter; however, for the sake of clear visualization it is presented as a separate server on the Zabbix map. The connection between SI Load Balancer and SI FTP is labeled as “SI FTP Adapter” and the connection line changes to a dashed red line if there is any responsiveness problem with the FTP adapter.

It was mentioned earlier that ORDERCREATION files are backed up on the Staging server. Monitoring consists of checking the timestamp of the last uploaded file and if there is no new file for two hours in the archive, an alert is raised.

Continuing the upper branch in Appendix 2 map - from SI FTP to FTP node - there is a monitoring item checking connectivity issue between these two nodes.

Finally, on the FTP node from where the external ERP systems pick the ORDERCREATION files up is a monitoring item checking for files older than two hours in dedicated folder. If there are such files in the folder, that indicates a problem with ERP downloading since normally ERP deletes the files from the FTP folder after they have been downloaded. In the shown case in Appendix 2 there is an *Information* severity alert about three such files displayed on the map.

A new connectivity checking item was created during this phase. Its purpose was to check the log files on the FTP server for any connectivity error to SI in the last two hours. An alert should be raised if such an error was found. A part of the script function is shown in the following code sample containing useful comments, i.e. the lines starting with ‘#’ symbol:

```
# get all error records from the last log file
```

```
GREP_RES=$(grep 'Errno' $NEWEST_LOG -B5)
# there is an error in the latest file -> get the last timestamp
if [[ $GREP_RES ]]; then
    # get all timestamps and tail the last one
    DATE=$(grep -Po "$DATE_REGEX" <<< "$GREP_RES" | tail -c23)
    TIMESTAMP=$(date -d "$DATE" +%s)
    # true -> last error occured in last 2 hours -> alert
    (($TIMESTAMP >= $NOW - $INTERVAL)) && { echo 1; } || { echo 0; }
else
    # no error in the latest file
    echo 0
fi
```

3.4.3 Dataload process visualizing

Dataload process is the most monitored WebSphere Commerce process consisting of multiple monitoring checks on most of the hosts the process flows through. The process Zabbix map can be found as Appendix 3. In many points in the flow, dataload can be perceived as “reverted“ Order creation in terms of direction of files‘ flow. Therefore, some checks – especially connectivity checks – are used in both flows.

All dataload files come from external ERP systems in the form of XML files. These files are first loaded to the FTP server to a store-specific input folder. In this moment, the first check takes place, iterating through all store-specific folders and counting the files. There is no trigger or alert configured for this check at the moment. Likewise, there is a check for the store-specific input folder size and file count. They serve to gather the data; however, no specific alerts are configured for them. These checks are not currently in use in Zabbix. The reason for this is that it has not yet been agreed in the team what the critical tresholds are.

If there are no new dataload files being uploaded to FTP during some specified time, this indicates a problem. In case of dataload files, there needs to be a check whether new files have come within the last two hours. If there are none, especially in production environment, an alert needs to be raised.

The connection between FTP and SI FTP adapter is monitored in the same way as in Order creation process.

Sterling Integrator, on the map represented by SI Load Balancer, performs checks of its FTP and HTTP (SOAP) interfaces and uses the same items which were already described in Order creation process.

Figure 10 displays a constant dataload cycle manipulating files from SI to Staging server. This process is active twice a day.

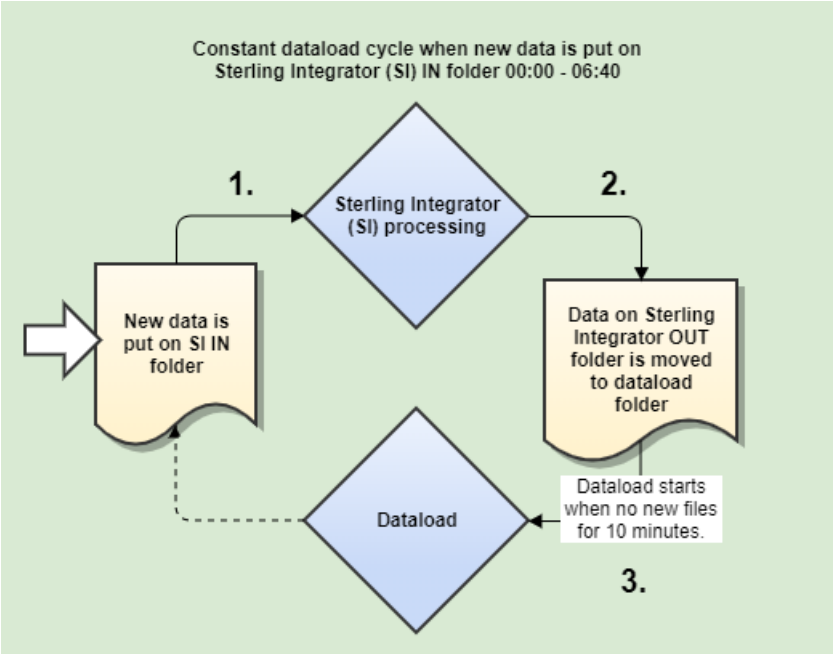


Figure 10. Dataload process on Sterling Integrator

The already processed CSV files in SI FTP are put to an */Out* folder to be moved to Staging server. Figure 10 describes this by the step number 2. The files should not reside in the */Out* folder for more than few minutes until they are moved to Staging server, thus the files should not be older as certain period of time: It was agreed that four hours is a good starting value for the monitoring item based on previous experiences.

The team required an alert if such files are found. The following code sample shows how this check is performed in the script. An `lftp` bash utility is used to access FTP via interface, and it performs a command getting the last uploaded file timestamp. This timestamp is tested whether or not it is over the specified 4-hour-value. For the purpose of this thesis, the credentials in `lftp` utility to connect to SI FTP are left out (they are provided as command line arguments to `lftp` utility with `-u` option).

```
OLDEST=$(lftp $IP/<path>/Out/$1 -e "ls -t | egrep -v '^d' | tail -n1;
bye" 2>/dev/null)
    if [[ $OLDEST ]]; then
        DATE=$(echo $OLDEST | cut -d ' ' -f 6-8)
        TIMESTAMP=$(date -d "$DATE" +%s)
        NOW=$(date +%s)
        # $2 configured in item key
        INTERVAL=$(( $2*60*60 ))
        (($TIMESTAMP <= $NOW - $INTERVAL)) && { RES=1; } || { RES=0; }
        echo $RES
    else
        echo 0
    fi
```

Figure 10 is a part of a bigger figure attached as Appendix 4 describing the Dataload and SOLR Indexing Schedules on a daily basis. The process described so far is shown as *1. phase* in Appendix 4, and the subsequent phases are explained further on.

Monitoring of Staging server, which is the next host in the dataload process flow, contains multiple existing and multiple new checks created during the process of visualizing.

After the data is processed by Sterling Integrator and put on the */Out* folder (Figure 10), there is a scheduled process that watches this folder and moves the files to Staging server's specialized store-specific dataload folder. It is important to be notified if files reside in this Staging server's folder for longer than 24 hours, since during this time they should be passed on further in the flow and hence, are not present in this folder anymore. Such a check is already implemented per each store-specific folder.

The dataload process itself, which performs loading data from received files to the Staging database (Figure 10 step number 3), is scheduled to run every time when there are no new files in the specialized dataload folder for ten minutes. In Zabbix there is a monitoring item which raises an alert when the dataload process is run. This item has only informational character; therefore the related trigger has *Information* severity.

During the process of visualizing the dataload process onto Zabbix map it was realized that an additional monitor should be added. In case the dataload process has not been run in the last 48 hours, this should be notified about. The monitor is performed by checking dataload folders for any new files in the last 48 hours.

The next step after the dataload process from Staging server to Staging database has been finished is to index the data. Indexing is a process of creating search indexes on Solr servers to enable searching of structured and unstructured data based on various search rules. These indexes are first created in the Staging environment and then later propagated to the production environment. In Dataload and Solr Indexing Schedules figure (Appendix 4) indexing is shown as *2. phase.* It can be observed that the indexing phase is happening twice a day, as well as other processes.

During the indexing, the first process is preprocessing the data which stores this data in a specific structure into temporary summary tables. After this is complete, the actual indexing takes place, extracting the data from the temporary tables and sending them to the Solr indexes. (Indexing external data in WebSphere Commerce search n.d.)

If this process fails or some partial errors or warnings occur during the process, this is logged into a store-specific logfile, since indexing takes place for every store separately. It is very important to monitor this process by monitoring the logfile and alerting if any problem occurs. Therefore, Zabbix monitoring item was created observing these logfiles and giving Zabbix the information about an exit code of indexing if it was not successful.

This indexing process is also displayed on Dataload Zabbix map (Appendix 3) as a connection between Staging server node and Solr Master node. In case any problem occurs, the connection line between these nodes indicates it visually by changing its color to red.

At this stage, all the data is ready for propagating to production servers, i.e. index propagation of indexes to Solr Repeater and Staging propagation of the actual data from Staging database to the production database. According to the figure in Appendix 4, this occurs twice a day always after the indexing has been finished. After this data is propagated to the production servers, it should be visible on the storefront

for buyers. Monitoring these processes is also important and therefore Zabbix checks were created for it. The propagation processes also create logfiles and indicate any error by the exit code in the logfile. Therefore, monitoring these processes is similar to monitoring indexing process.

On the Zabbix map (Appendix 3), index propagation and Staging propagation are displayed as connections between regarded nodes. For even better monitoring of connection between Solr Master and Solr Repeater, a new connectivity check was created.

The search data, however, is not usable in production until the final step of propagation, namely propagating to Solr Slaves is done. These nodes are shown on the Zabbix map (Appendix 4) as the last ones on the right and connectivity checks from Solr Repeater were implemented here as well. The reason for using the Solr Slaves is that there are two WCS Application server nodes in use in case any downtime on one of them occurs. Each WCS Application server has its own Solr Slave server to use search index from.

To finalize the propagation process, it must be mentioned that the product images visible on the storefront do not undergo this whole process. They are propagated without any indexing as the bottom part of the figure shown in Appendix 4. Monitoring of this part has not been implemented yet; however, it has been realized that it must be discussed upon in the future since this is also a business critical process.

4 Monitoring transfer to SolarWinds Orion Platform

4.1 Transfer specification

The creation of visualizing business critical processes on Zabbix map fulfilled its purpose. New monitoring needs have arose and author's work has been accomplished in this part by satisfying these needs and creating sufficient monitoring checks. The next step towards finalizing the requirements of the customer to transfer current monitoring solution to SolarWinds platform must be taken. Before the

transfer is in progress, technical specification of what exactly needs to be transferred and how is it supposed to be done needs to be created. The specification will also serve as detailed documentation of current setup of Zabbix and will be shared internally in author's team.

The specification is done in the form of an Excel workbook. The concept is to document and specify all used Zabbix checks with all the needed data so that there is a base for the SolarWinds transfer.

As mentioned earlier, the Zabbix items are logically grouped into Zabbix templates. These templates are then applied on configured hosts, meaning all the Zabbix items in the template are applied on respective hosts. Therefore, each template is going to be represented by one Excel table. Example of one of the tables can be found as Appendix 5.

Every table defines which hosts are grouped in it in its header, hence which hosts contain items mentioned in the table. First column of each table displays alert text related to second column value displaying alert trigger expression. Subsequently, these values are related to third column – the Zabbix item itself. Additional columns are introducing parameters to item if they exist, then update interval in seconds of associated item is shown. The last columns show whether the item or the trigger is enabled per each host. Coloring of these cells is explained in an extra table.

If the cell in *Alert* or *Item* columns is gray, that means the item or the trigger is disabled on template, which means that Zabbix is not gathering any data for such item, or is not evaluating the trigger expression for such trigger. The reasons for disabling an item or trigger are various. For instance, if the item (trigger) served only for testing purposes when the monitoring check was being created or when the item (trigger) is currently not in use, it is prefered to disable the item (trigger) over deleting it, as it might be used again in the future. If such "template-level" disability is set on an item or a trigger, all the hosts configured on this template have the item or the trigger disabled by default as well. However, it is possible to overwrite this on the "host-level" and enable the item or the trigger only there.

The last columns are representing the status of an item and a trigger respective to each host. The values in the last columns can be colored green which indicates that

related item is enabled on host, or it can contain one of the values *'D'* - disabled on host, *'NS'* – not supported on host, *'U'* – unknown on host, *'NE'* – not existing on host. Exceptionally, value and color in these columns showing trigger status can represent trigger severity, which overwrites the default "template-level" severity on related host.

There are few entries in the table in Appendix 5 requiring further explanation. In the original Excel sheet there is an additional table consisting of all the necessary notes and explanations, so that the reader can refer to it when any information is not clear.

In the Appendix 5 there is one item named *'_checkJenkinsService'*. The convention is that if an item is prefixed with the underscore symbol '_', this indicates that it is not part of the team's custom checks. The custom checks are represented as a function in the monitoring.sh wrapper script and represent the user parameter Zabbix checks. In contrary, items prefixed with the underscore symbol '_' are either Zabbix default keys or one-liner user parameter agent-performed checks. Concretely, this *'_checkJenkinsService'* item uses Zabbix default `net.tcp.service[tcp,,8080]` key.

As described earlier, the *'DB Connection error'* trigger and the *'checkDatabase'* and *'checkDataloadPremergeAllXmlFileCount'* items are disabled on template.

The *'checkStuckOrdersMStatusV8'* item shows an example of trigger severity being overwritten. The "template-level" severity is *Not classified* as indicated by the grey coloring of the trigger expression, however the columns representing state of the trigger per each host indicate that for Production Staging server the severity is set to *Average* as indicated by the orange color and the text *'Aver'* in associated cell. Similarly, for QA staging server the severity is set to *Warning* as indicated by the yellow color and the text *'Warn'* in an associated cell.

Lastly, there are two items, concretely *'checkStagingprop'* and *'checkIndexing[]'*, which have a value *'flexible'* in the *Update* column. The Excel spreadsheet explains what this flexible value means. These items do not have a default update interval, but rather there is specified a period of time in the week (given by information about the days in the week and range of hours) and update interval applied when the item is active. An entry saying `Interval: 60; Period 1-7, 11:30-11:31;1-7, 23:30-23:31` indicates that the item is gathering the data twice every day in specified periods.

4.2 Current situation in SolarWinds

After creating the specification for the transfer another step is to begin implementing the transfer itself. It is important to mention that the author's team customer has been using the SolarWinds monitoring solution already in their processes and the monitoring implementation for the author's AMS team is going to be partly under the control of the customer. Therefore, the author's team is given only user level privileges to the SolarWinds platform and all the configuration must be done by ordering it from SolarWinds team. This gives more security control over the monitoring for the customer and the SolarWinds team. However, it brings constraints for the AMS team to perform the whole transfer easily and conveniently.

As mentioned earlier in this thesis when introducing the SolarWinds Orion platform, SolarWinds comes with some monitoring templates "out of the box". The templates are mainly focused on CPU utilization of monitored nodes, memory usage and disk space occupancy. These templates were applied on all the configured nodes together with an alert conditions coming "out of the box". In table 1 there is an overview of what comes with the platform known as *Global alerts.*

Table 1. SolarWinds Global Alerts

Alert name	**Alert condition**
Global Memory	Above 90% utilization longer than 15 minutes
Global CPU	Above critical threshold of 90% for 15 minutes
Global Volume	Space size < 5%/3%/1% on volumes < 1TB/10TB/ > 10TB
Global Node Up/Down	Node is not responding for 5 minutes
Global High packet loss	Percent packet loss over the last few minutes between 5% and 40%

Alert severities

When it comes to severities of alerts, SolarWinds does not offer as flexible configuration. There are mainly three severity levels – *Information*, *Warning* and *Critical*. The Zabbix platform offered flexible configuration and the author's team used five severity levels for all the alerts. Therefore, the alert severities need to be adjusted to the

new standards when later implementing the custom monitoring. All the Global alerts have *Critical* severity level.

Implemented applications

Before the author was given an access to the SolarWinds platform and he started working on the transfer himself, some monitoring had already been implemented in the platform in the past. This had been done when the customer had first started using the platform for his processes and had decided that the transfer for author's AMS team will be requested. The servers (nodes) were configured to be polled and some monitoring applications were created.

Applications are the containers for individual monitoring components. They are equivalent to templates in Zabbix. They can be assigned to nodes which are then monitored by the application components. Components are equivalent to items in Zabbix.

There are four already existing applications created. Following Tables 2-5 show the application components as well as the nodes where the applications are configured. Components that are identical or similar to items implemented in Zabbix are marked with yellow background color.

1. HADR Health Application. This application monitors a database High Availability Disaster Recovery (HADR) status. This indicates when there is any problem with primary database and secondary database needs to take control. Custom monitoring for this is setup also in Zabbix.

Table 2. HADR Health

Nodes	**Components**
DB servers	HADR Role
	HADR Connect Status
	HADR Heartbeat
	HADR Log Gap
	HADR State
	HADR Sync Mode

2. Linux CPU Monitoring Application. CPU related monitoring in Zabbix offers few more customized items and triggers compared to the ones below, such as monitoring of process load. That needs to be added to SolarWinds platform.

Table 3. Linux CPU Monitoring

Nodes	**Components**
All servers (except SI servers)	CPU User Time (%)
	CPU System Time (%)
	Wait IO (%)
	CPU Idle Time (%)
	Run queue
	Interrupts per second
	Context switches per second
	Total amount of interrupts after boot
	Total amount of CPU context switches after boot

3. Linux Memory Monitoring Application. Additional item currently in Zabbix but not in SolarWinds is monitoring free swap space in percentage and triggering an alert if it drops under 50%.

Table 4. Linux Memory Monitoring

Nodes	**Components**
All servers (except SI servers)	Total memory (kB)
	Used memory (kB)
	Free memory (kB)
	Total swap (kB)
	Used swap (kB)
	Free swap (kB)
	Buffers (kB)
	Cache (kB)
	Dirty Pages (kB)
	Anonymous Pages (kB)
	Amount of zombie processes

4. DB Monitoring Application. Monitoring of databases in this fashion is an extra value brought by SolarWinds. In Zabbix all database monitoring was custom.

Table 5. DB Monitoring

Nodes	Components
DB servers (except SI DBs)	Database Used Space (MB)
	Log File Used Space in Specified Database (MB)
	Log File Free Space in Specified Database (MB)
	Average Buffer Total Hit Ratio (%)
	Average Data Hit Ratio (%)
	Average Index Hit Ratio (%)
	Number of Locks Held in Specified Database
	Average Read Time (ms)
	Connected applications to Specified Database
	Number of Long Running Queries
	Number of Table Scans
	Table with the Biggest Table Scans Value
	Used Space of the Biggest Table (MB)

No triggers or alerts have been associated with these components yet. The specification is included in the Excel workbook dedicated for that.

4.3 Ordering a sample monitoring in SolarWinds

So far only the basic monitoring was described and plans for how to proceed with its transfer were shown. Next step is to show how the custom monitoring will be transferred.

It was decided in the team that the first step is to order a sample monitoring to see how fast and flexible the SolarWinds team is in implementing the solution. Three monitoring items and alerts were chosen to be created with different level of implementation difficulty from easy to implement to complex check. The checks were decided to be made only on the test servers for now. The items are

1. Kernel maxfiles. This monitor belongs to the basic monitoring because it utilizes Zabbix default item key. It checks for configured Linux maximum of opened files and fires an alert if this configuration drops under a threshold. This item should be configured on all servers.
2. Dataload process. This belongs to the custom monitoring configured as user parameter in Zabbix. Dataload process runs only on Staging servers and therefore this check should be performed there.

3. Check old dataload files in SI shop-specific folder. This check is considered to be complex because it runs the wrapper script function on SI node. The script function connects to SI FTP and in every shop-specific folder it finds the oldest file. Then it checks if this file is older than 4 hours. If that happens, an alert should be fired.

The wrapper script `monitoring.sh` used in Zabbix is going to be utilized also for SolarWinds. The components will run the script function and evaluate the output. Linux script monitoring in SolarWinds requires the script to output value in predefined format. The function should output the value in the following fashion:

```
#monitoring.sh
#!/bin/bash
function monitoringFunction()
{
    # body of the script
    echo "Message.1: $message1"
    echo "Statistics.1: $statistics1"
    echo "Message.2: $message2"
    echo "Statistics.2: $statistics2"
    exit 0
}
```

The variables `message1, statistics1, message2, statistics2` are the values recognized by SolarWinds evaluator. The exit codes indicate the monitor status. Exit codes have predefined meanings, as shown in table 6. (Linux/Unix Script Monitor n.d.)

Table 6. Linux script functions exit codes meaning in SolarWinds (Linux/Unix Script Monitor n.d.)

Exit Code	**Meaning**
0	Up
1	Down
2	Warning
3	Critical

Therefore, `monitoring.sh` script is modified to fulfill SolarWinds standards.

4.4 Implementing sample monitoring

The job that had to be done on the SolarWinds site was to follow the Excel documentation and with its help implement the sample monitors. First of all, the application template needed to be created and assigned to the correct nodes. Secondly, the sample components needed to be created on the template. Using the SolarWinds application template is equally beneficial as using Zabbix template, since it increases the maintainability of the monitoring by applying configuration to all nodes in a batch. There is no need to configure the same parts of monitoring per each node separately.

The way how SolarWinds SAM platform executes these Linux checks is either via an agent installed on the servers or via SSH. Therefore, every node needs to be configured with credentials for SSH login. Linux *solarwinds* user was created on each node and was assigned required privileges to be able to run the `monitoring.sh` script.

Figure 11 shows the application overview of one of the sample monitors in the SolarWinds application tree, i.e. the dataload process monitoring. For security reasons, the DNS name of each node was left out.

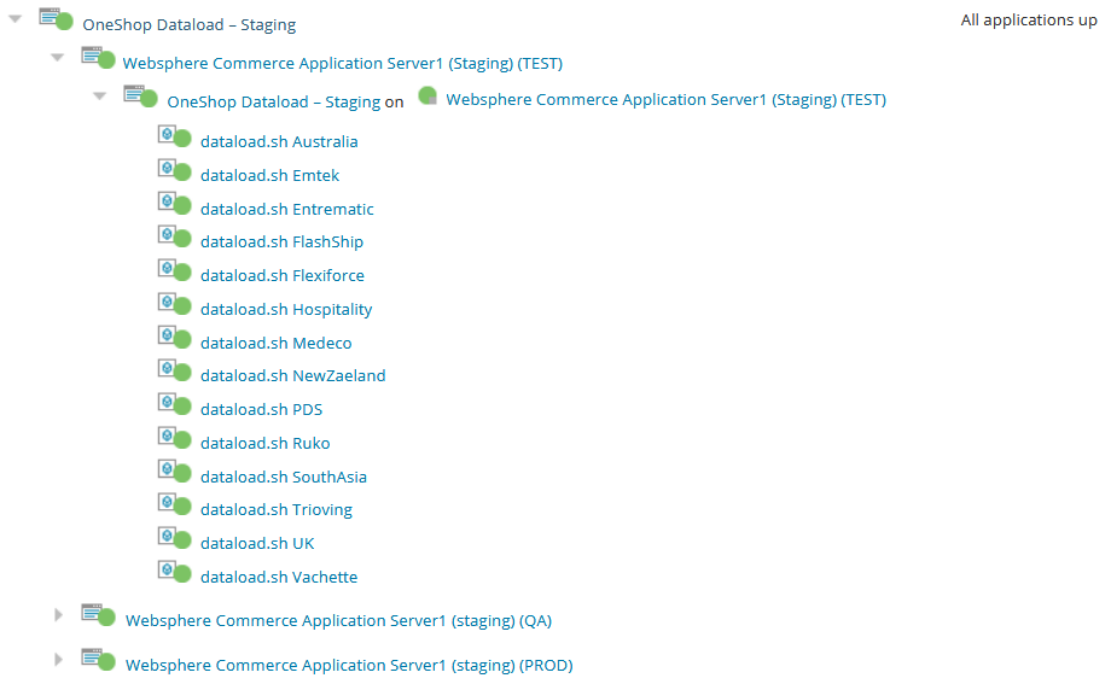


Figure 11. Sample monitoring of dataload process in SolarWinds

The application template is applied also for the bottom nodes *Websphere Commerce Application Server1 (staging) (QA)* and *Websphere Commerce Application Server1 (staging) (PROD).* That means the components *dataload.sh <store>* are configured the same way on all three nodes.

The SolarWinds Orion platform offers much more high-level statistics and graphs about applications and individual components. Figure 12 displays a graph of history statistics of monitored value for one component from the dataload monitoring application. This helps to identify in which time or intervals the component increases its value and helps to make long-term predictions.

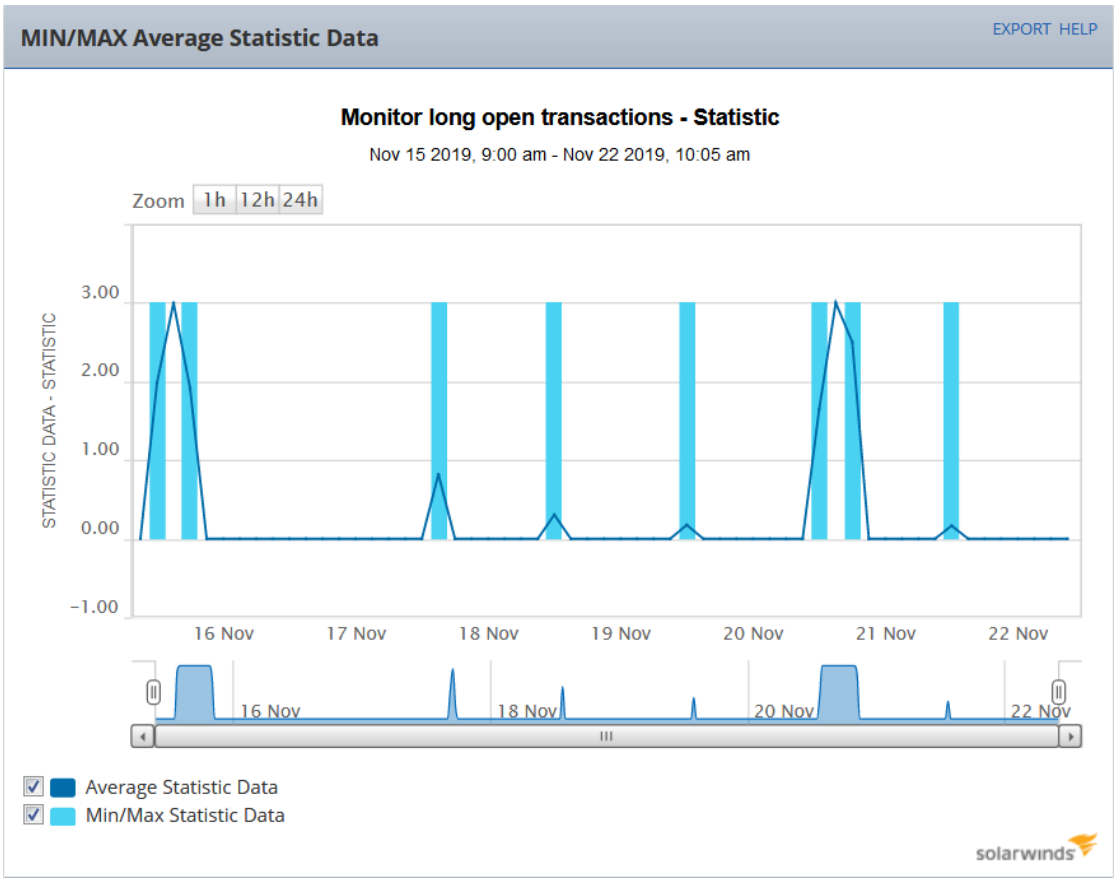


Figure 12. Dataload process monitor component history statistics

The next step in implementing the monitoring is to create an alert for the application components. For the dataload process monitoring two alerts were created

1. The first alert fires when the dataload process starts and, similarly as in Zabbix, has only informational character

2. The second alert has warning severity and fires when the dataload process runs for longer than 60 minutes which may indicate a problem.

All the alerts are displayed on the alert dashboard and will be the most valuable information for the AMS team. Figure 13 shows such dashboard with some Global alerts and one dataload alert.

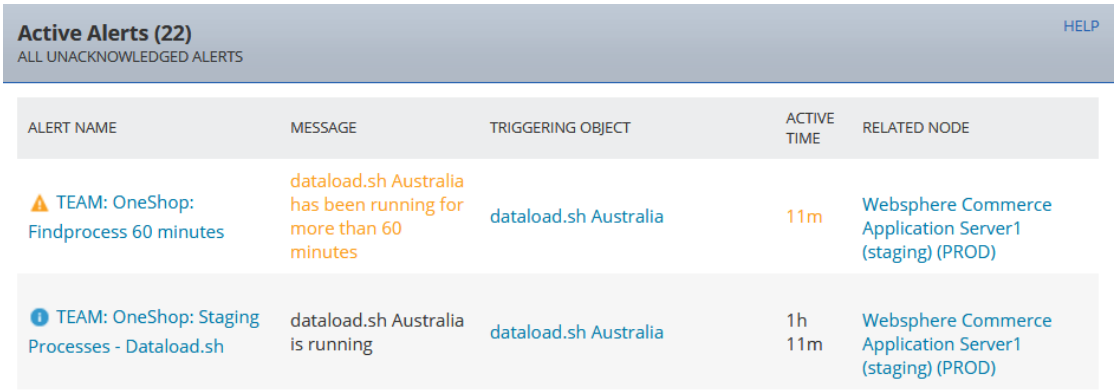


Figure 13. SolarWinds alert dashboard

5 Conclusion

5.1 Summary

Network and application monitoring are constantly developing. The monitoring metrics are being advanced, performance of monitoring is being improved and visual presentation such as graphs maps or dashboards are being adjusted to modern trends. Naturally, monitoring platforms are trying to accommodate to these trends and doing all they can to remain competitive on the market. Software companies utilizing monitoring choose the platform which best suits their needs and fulfills the latest standards. Often, the transfer of the monitoring needs to be performed if it is realized that some other platform brings better value to the team. Such transfer was implemented in this thesis.

The first step was to get familiar with the current monitoring solution in Zabbix platform. I started to dig deeper in the configurations to understand the system. New monitors were required to be implemented and old configuration was needed to be

reviewed. During this process, internal documentation about the monitoring was improved and enhanced.

Visualizing of the monitoring was a new task for the team and I needed to understand how it works in Zabbix and visualize the most critical processes of the application on a network map. These maps help the team to identify problems quicker and more conveniently. It also helped me to grasp even better understanding of the processes which are being monitored. This initiated creative process of finding out critical pieces of the application which were not being monitored and such monitoring was implemented.

The next step was to perform the actual transfer to SolarWinds monitoring platform. The specification of how the transfer should be done was created and sample monitoring was ordered from SolarWinds.

5.2 Transfer continuation

Unfortunately, the SolarWinds team was not so agile and flexible in implementing this sample monitoring. It was assumed that the sample monitors will be done in shorter time as they in fact were. The SolarWinds team has been working on multiple projects at the same time and our task was not majorly prioritized. The fact that the AMS team did not have any access to editing the configuration on the SolarWinds site made the whole process very slow.

Therefore, it was agreed in the team that privileges to edit the monitoring will be requested from SolarWinds to some extent. The SolarWinds team assigned edit privileges on application templates and components for the team. Although, they could not assign alert edit privileges for security reasons, since the privileges to edit alerts can be given only per the whole platform and the platform is used by many other customers. It means that the creation of alerts stays as the responsibility of SolarWinds site.

By the time this thesis has been concluded, the sample monitoring has been implemented and some additional monitoring applications and components have been configured. The issues with connection to nodes via SSH have occurred seldom. For

instance, the components in an application template *SI general* experienced a problem as shown in Figure 14.

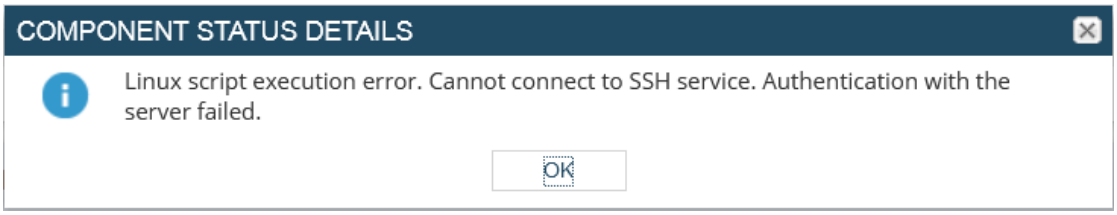


Figure 14. Linux script execution error

Nevertheless, it was tested that the Linux script execution does not fail but the component fails for some reason anyway. This issue has been discussed on SolarWinds site and the cause of the error is still unknown.

However, since the SolarWinds has implemented the sample monitoring, the process became more familiar to both the AMS and SolarWinds site and further development of monitoring is becoming faster. The transfer will be continued until all the current monitoring in Zabbix is transferred. Then, testing period will last for few months on both platforms and the Zabbix platform will be finally decommissioned after successful testing.

References

Agent vs. Agentless Monitoring. 2016. Page on ScienceLogic website. Accessed on 09 April 2019. Retrieved from https://sciencelogic.com/blog/agent-vs-agentless-monitoring

Application monitoring. N.d. Page on Techopedia website. Accessed on 02 December 2019. Retrieved from https://www.techopedia.com/definition/29133/application-monitoring

Basics of Network Monitoring. N.d. Page on ManageEngine website. Accessed on 20 March 2019. Retrieved from https://www.manageengine.com/network-monitoring/basics-of-network-monitoring.html

Bigelow, S. 2018. *Guide to buying server performance monitor software.* Page on TechTarget website. Accessed on 10 March 2019. Retrieved from https://searchdatacenter.techtarget.com/tip/Guide-to-buying-server-performance-monitoring-software

Indexing external data in WebSphere Commerce search. N.d. Page on IBM Knowledge Center website. Accessed on 7 October 2019. Retrieved from https://www.ibm.com/support/knowledgecenter/en/SSZLC2_8.0.0/com.ibm.commerce.tutorials.doc/tutorial/tsd_search2_intro.htm

Integrating. N.d. Page on IBM Knowledge Center website. Accessed on 15 May 2019. Retrieved from https://www.ibm.com/support/knowledgecenter/SSZLC2_8.0.0/com.ibm.commerce.integration.doc/concepts/ccvcapabilities.htm

Jenkins (software). N.d. Page on Wikipedia website. Accessed on 23 September 2019. Retrieved from https://en.wikipedia.org/wiki/Jenkins_(software)

Kitili, G. 2018. *What is application maintenance.* Page on Quora website. Accessed on 14 May 2019. Retrieved from https://www.quora.com/What-is-application-maintenance

Korzeniowski, P. 2016. *Agent vs. agentless: Monitoring choices for diverse IT ops needs.* Page on TechTarget website. Accessed on 10 March 2019. Retrieved from https://searchitoperations.techtarget.com/tip/Agent-vs-agentless-Monitoring-choices-for-diverse-IT-ops-needs

Linux/Unix Script Monitor. N.d. Page on Solarwinds documentation website. Accessed on 15 November 2019. Retrieved from http://www.solarwinds.com/documentation/en/flarehelp/sam/content/sam-linux-unix-script-monitor-sw3260.htm?cshid=OrionAPMPHComponentTypesLinuxScript

Network monitoring. N.d. Page on Wikipedia website. Accessed on 14 March 2019. Retrieved from https://en.wikipedia.org/wiki/Network_monitoring

Olups, R. 2019. *Zabbix 4 Network Monitoring.* 3rd ed. Birmingham: Packt Publishing Ltd.

One platform to rule your IT stack. N.d. Page on solarwinds website. Accessed on 22 November 2019. Retrieved from https://www.solarwinds.com/solutions/orion.

Rouse M. 2018. *Zabbix.* Page on TechTarget website. Accessed on 22 March 2019. Retrieved from https://searchitoperations.techtarget.com/definition/Zabbix

Server&Application Monitor. N.d. Page on SolarWinds Documentation website. Accessed on 8 November 2019. Retrieved from https://documentation.solarwinds.com/en/Success_Center/SAM/Content/SAM-Using-Application-Monitor-Templates-sw1115.htm

SolarWinds. N.d. Page on Wikipedia website. Accessed on 8 November 2019. Retrieved from https://en.wikipedia.org/wiki/SolarWinds

SolarWinds Orion Platform Integration. N.d. Page on Notepage website. Accessed on 8 November 2019. Retrieved from https://www.notepage.net/solar-winds/orion.htm

Staging server. N.d. Page on IBM Knowledge Center website. Accessed on 21 August 2019. Retrieved from https://www.ibm.com/support/knowledgecenter/en/SSZLC2_7.0.0/com.ibm.commerce.admin.doc/concepts/csstagingserver.htm

Sterling B2B Integrator. N.d. Page on IBM Knowledge Center website. Accessed on 30 September 2019. Retrieved from https://www.ibm.com/support/knowledgecenter/en/SSVSD8_8.4.1/com.ibm.websphere.dtx.md.doc/topics/g_md_sb2bi_sterling_b2b_integrator.htm

WebSphere Commerce Version 8. N.d. Page on IBM Knowledge Center website. Accessed on 14 May 2019. Retrieved from https://www.ibm.com/support/knowledgecenter/SSZLC2_8.0.0/landing/wc_welcome.html

Wilson M. 2019. *Best Network Monitoring Tools & Software of 2019*. Page on pcwdld website. Accessed on 22.05.2019. Retrieved from https://www.pcwdld.com/best-network-monitoring-tools-and-software

Zabbix agent. 2018. Page on Zabbix Documentation website. Accessed on 10 April 2019. Retrieved from https://www.zabbix.com/documentation/4.0/manual/concepts/agent

Zabbix configuration. 2018. Page on Zabbix Documentation website. Accessed on 12 April 2019. Retrieved from https://www.zabbix.com/documentation/4.0/manual/config

Zabbix discovery. 2018. Page on Zabbix Documentation website. Accessed on 12 April 2019. Retrieved from https://www.zabbix.com/documentation/4.0/manual/discovery

Zabbix proxy. 2018. Page on Zabbix Documentation website. Accessed on 10 April 2019. Retrieved from https://www.zabbix.com/documentation/4.0/manual/concepts/proxy

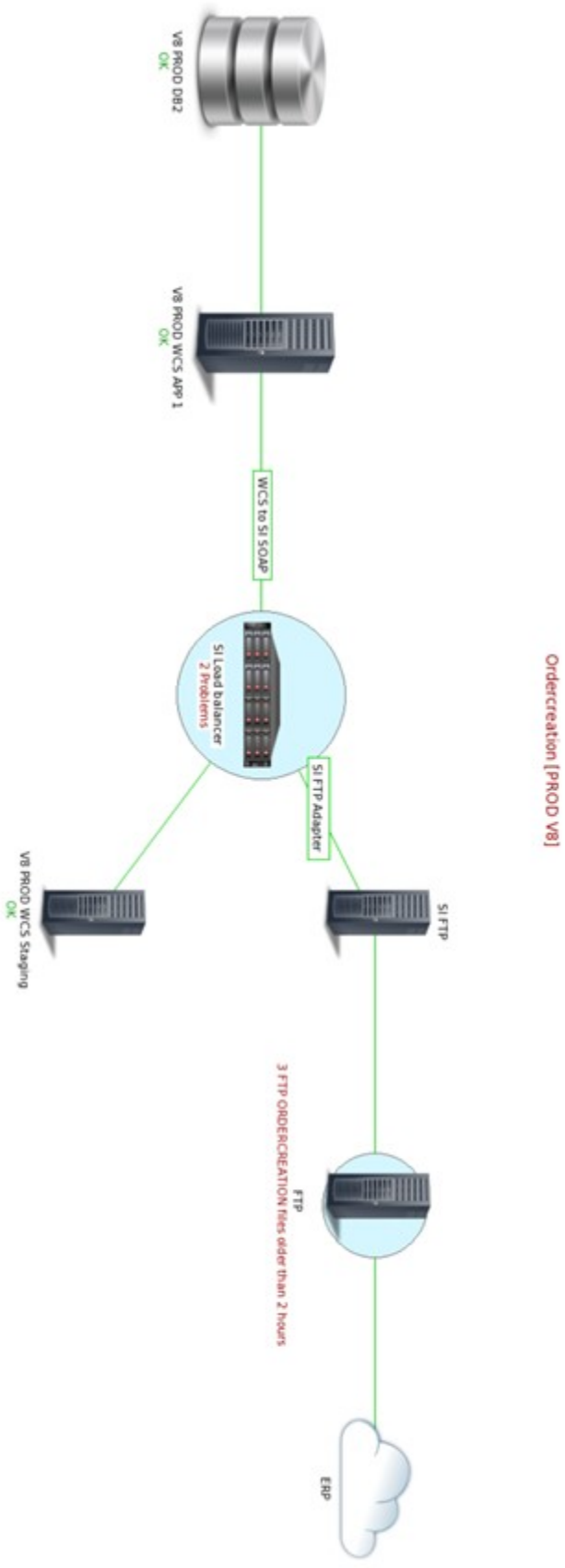
Zabbix visualisation. 2018. Page on Zabbix Documentation website. Accessed on 10 April 2019. Retrieved from https://www.zabbix.com/documentation/4.0/manual/config/visualisation

Appendices

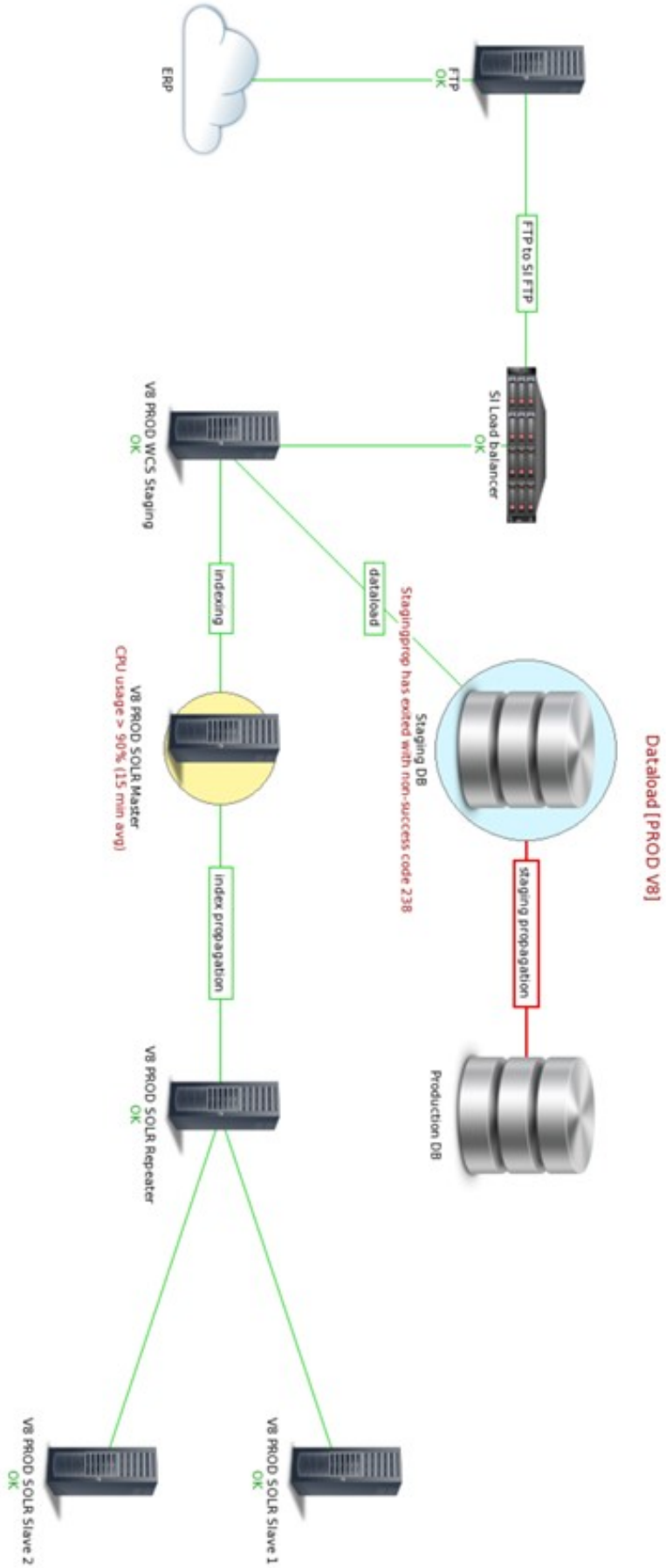
Appendix 1. Example of Monitoring documentation

Monitored item/service	Purpose & Description	Item in Zabbix	Template	Trigger(s) and severity	Actions when triggered	Hosts where this needs to be monitored on
Failure of database backups	Database backup script creates an error file if the database backup has been unsuccessful for some reason. Item checks if there are these error files in the log folder that are newer than 4 days. Run by Zabbix agent, configured to run function checkDbBackupStatus() in monitoring.sh script file.	checkDbBackupStatus	Template monitoring.sh - DB2 Backup Status	**INFORMATION:** DB BACKUP ERROR - {ITEM.VALUE} backup error files found	DB2 backup script has failed to do a backup and has not removed older backup files. Investigate backup log file created by the script why this has happened if additional actions are required. Possible reasons include DB not up, filled disks. If there are successful backups after the failed one, most likely no further action is required.	All hosts that run DB2 and backups according to backup plan. Bolded servers have been verified: **PROD DB2 Primary** PROD DB2 Secondary **QA DB2 Primary** QA DB2 Secondary PROD SI DB2 1 PROD SI DB2 2 QA SI DB2 1 QA SI DB2 2
Jenkins service not running	Get to know when Jenkins service stops working Runs on QA Staging (V7, V8) Using zabbix item key net.tcp.service[tcp,,8080]	net.tcp.service[tcp,,8080]	Template monitoring.sh - staging WiPro	**WARNING:** Jenkins service is not running	Verify if Jenkins service is not running, eventually restart.	V8 QA WCS Staging QA WCS Staging
SI FTP return code	Item checks for SI FTP return code. Run by Zabbix agent, configured to run function checkSIFTPHTTPCode() in monitoring.sh script file.	checkSIFTPHTTPCode	Template SI general WiPro	**WARNING:** Sterling Integrator FTP - wrong response code		All SI hosts: TEST SI QA SI 1 QA SI 2 PROD SI 1 PROD SI 2

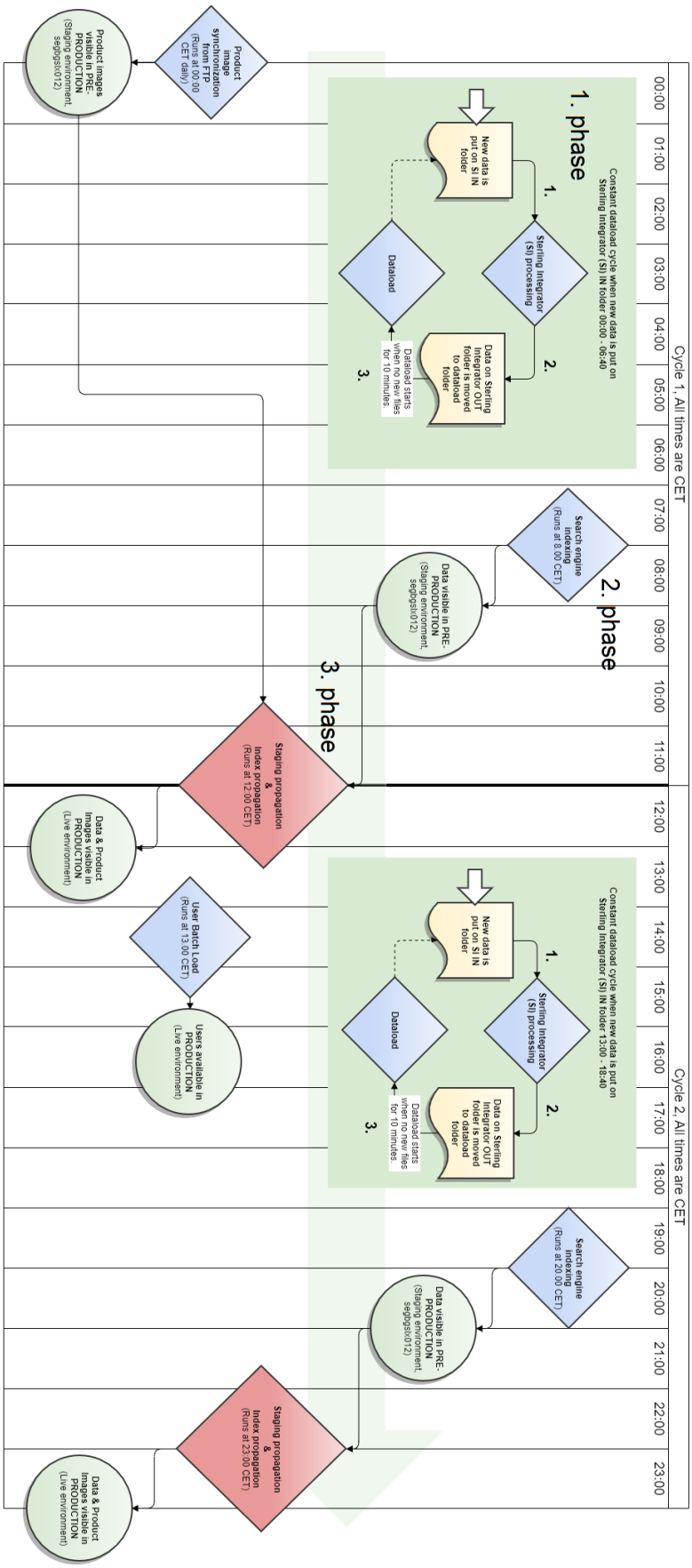
Appendix 2. Order creation process Zabbix map



Appendix 3. Dataload process Zabbix map



Appendix 4. Dataload and SOLR Indexing Schedules



Appendix 5.

Alert	Trigger expression	Template	param	Update	Item	Trig	Item	Trig
		Staging Monitoring			0081		0069	
		Hosts						
		V8 PROD Staging dm0081						
		V8 QA Staging dm0069						
		Items						
DB Connection error	Item.last()<>0	checkDatabase		60	D		D	
		checkDataloadPremergeAllXmlFileCou		60	D	NE	D	NE
CSV files older than 24h found in pre-mergedata <param>	item.now() - item.last()>86400	checkDataloadPremergeOldestFile[]	Accessories	30				
CSV files older than 24h found in pre-mergedata <param>	item.now() - item.last()>86400		Aptus					
CSV files older than 24h found in pre-mergedata <param>	item.now() - item.last()>86400		Vachette					
Real time pricing service error from WCS to onlineoe	item.last()=0	checkIsRealTimePricingUp[]	onlineoe	60			NS	U
Real time pricing no data for 15 mins from ooe to WCS	item.sum(900)=0		ooe				NS	U
{ITEM.LASTVALUE} order(s) in OneShop V8 stuck in status M for more than 10 minutes!	item.last()>0	checkStuckOrdersMStatusV8		600		Aver		Warn
NO NEW WCS ORDER CREATION for 2 hour (by count)	item.delta(2h)=0	checkWCSOrderCreationCount		60			D	D
NO NEW WCS ORDERCREATION files for 2 hour (by timestamp)	item.now() - item.last()>=7200	checkWCSOrderCreationTimestamp		60			D	D
Jenkins service is not running	item.last()=0	_checkJenkinsService		1800	D	D		
Stagingprop has exited with non-success code {ITEM.LASTVALUE}	item.last()>0	checkStagingprop		flexible				
HTTP1 server is down	item.sum(#2)=0	checkHTTPService[]	IHS1	30				
HTTP2 server is down	item.sum(#2)=0		IHS2					
Indexing for <param> has exited with non-success code {ITEM.LASTVALUE}	item.last()>0	checkIndexing[]	accessories	flexible				
Indexing for <param> has exited with non-success code {ITEM.LASTVALUE}	item.last()>0		aptus					
Indexing for <param> has exited with non-success code {ITEM.LASTVALUE}	item.last()>0		vachette					
Connection from Staging WCS to SOLR Master not working	item.last()=0	checkConnectionSOLRMaster		60				