

Zabbix meets television

Clever use of Zabbix features



IntelliTrend GmbH

www.intellitrend.de



Contact: Wolfgang Alper

wolfgang.alper@intellitrend.de



ZDF – Zweites Deutsche Fernsehen

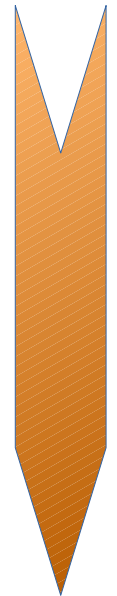
ZDF - „Second german television“ Some history

ZDF - History

- In **1961**, the federal states established a central **non-profit** television broadcaster „**Zweites Deutsches Fernsehen**“.
- In **1963** on April 1, ZDF officially went on air and had reached 61 percent of television viewers.
- On the Internet, a selection of programs is offered via live stream or video-on-demand through the „ZDFmediathek“, which has been in existence since **2001**.
- Since February **2013**, ZDF has been broadcasting its programs around the clock as an Internet livestream.
- As of today ZDF is one of the largest public broadcasters in Europe with permanent bureaus worldwide, and is also present on various platforms like youtube, facebook etc.



1961



2021

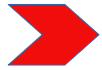
ZDF – Monitoring with Zabbix

Let's get technical Some clever uses of Zabbix features



A special thanks goes to Mr. Uwe Grunert for the good joint work!

ZDF – Monitoring with Zabbix



Get states from infrastructure monitoring with dynamic severities using LLD

Goal

- Monitor and alert data coming from an external monitoring system that controls infrastructure components such as power generators, transmission stations and the like.
- The external system should automatically define the services to be monitored in Zabbix.
- The external system should automatically define the trigger severity levels to be used in Zabbix for each service.

Challenges

- How can the external system automatically define the severity levels to be used by Zabbix triggers?



ZABBIX
PREMIUM PARTNER

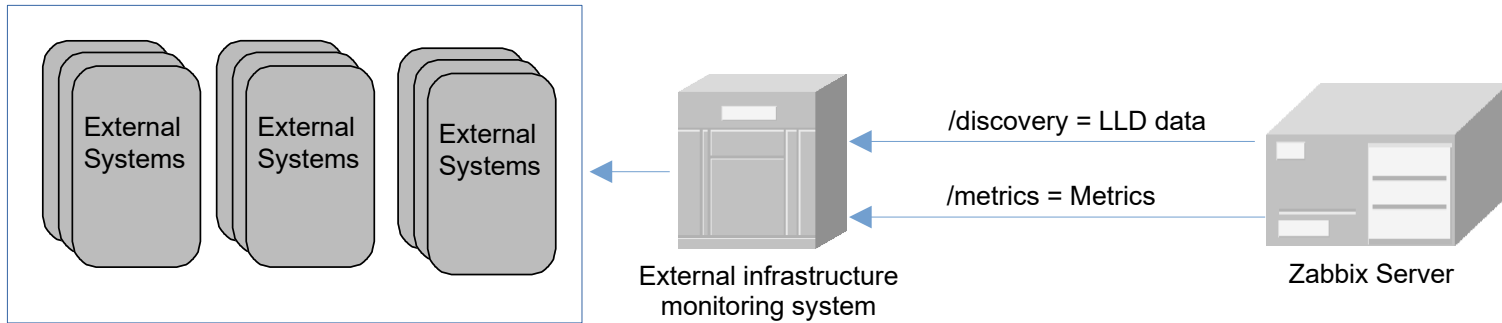
ZDF – Monitoring with Zabbix



Get states from infrastructure monitoring with dynamic severities using LLD

Approach

- Use Zabbix build in HTTP check to get LLD discovery data.
- Use Zabbix build in HTTP check as a collector get metrics.
- Define item prototypes as dependant items to extract data from collector item.
- Create „smart“ trigger prototypes to respect severity information from LLD discovery data.



Note: HTTP item allows to use zabbix-sender, which is great for testing.

ZDF – Monitoring with Zabbix



Get states from infrastructure monitoring with dynamic severities using LLD

```
{
  "{#NAME}": "generator-secondary",
  "{#DISPLAYNAME}": "Secondary power generator",
  "{#DESCRIPTION}": "Secondary emergency power generator",
  "{#CATEGORY}": "Powersupply",
  "{#PRIORITY.INFORMATION}": -1,
  "{#PRIORITY.WARNING}": -1,
  "{#PRIORITY.AVERAGE}": -1,
  "{#PRIORITY.HIGH}": 1,
  "{#PRIORITY.DISASTER}": 2
}
```

LLD returned by /discovery

Each component defines its assignment from the „status“ value to a specific severity level.

A value of -1 means: not used.

Metrics returned by /metrics

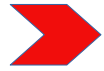
In this example, „status“ = 1 refers to a priority of {#PRIORITY.HIGH}.

```
"generator-primary": {
  "status": 0,
  "message": "Generator is healthy."
},
"generator-secondary": {
  "status": 1,
  "message": "Generator is not working properly."
},
```

status = 0 means, no issues.



ZDF – Monitoring with Zabbix



Get states from infrastructure monitoring with dynamic severities using LLD

Item prototypes

Planit										Enabled	ZBX	Discovery list / Planit Discovery		Item prototypes 2	Trigger prototypes 5	Graph prototypes	Host prototypes		
rd	Name ▲																		
	TPL Planit: Planit Collector: {#DISPLAYNAME} [message]																		
	TPL Planit: Planit Collector: {#DISPLAYNAME} [status]																		

Preprocessing 1

steps	Name	Parameters
1:	JSONPath	<code>\${#NAME}["message"]</code>

Add

All templates / TPL Planit / Discovery list / Planit Discovery / Item prototypes 2 / Trigger prototypes 5 / Graph prototypes / Host prototypes

Item prototype Tags 1 Preprocessing 1

* Name

{#DISPLAYNAME} [message]

Type

Dependent item

* Key

message[{#NAME}]

Select

* Master item

TPL Planit: Planit Collector

Select

Type of information

Character

* History storage period

Do not keep history

Storage period

90d

Value mapping

type here to search

Select

Description

{#DESCRIPTION}



ZDF – Monitoring with Zabbix



Get states from infrastructure monitoring with dynamic severities using LLD

Trigger prototypes

All templates / TPL Planit Discovery list / Planit Discovery Item prototypes 2 Trigger prototypes 5 Graph prototypes Host prototypes						
<input type="checkbox"/>	Severity	Name ▲	Operational data	Expression	Create enabled	Discover
<input type="checkbox"/>	Average	planit Service: {#DISPLAYNAME} has Status AVERAGE		<code>last(/TPL Planit/status[{#NAME}]) = {#PRIORITY.AVERAGE}</code>	Yes	Yes
<input type="checkbox"/>	Disaster	planit Service: {#DISPLAYNAME} has Status DISASTER		<code>last(/TPL Planit/status[{#NAME}]) = {#PRIORITY.DISASTER}</code>	Yes	Yes
<input type="checkbox"/>	High	planit Service: {#DISPLAYNAME} has Status HIGH		<code>last(/TPL Planit/status[{#NAME}]) = {#PRIORITY.HIGH}</code>	Yes	Yes
<input type="checkbox"/>	Information	planit Service: {#DISPLAYNAME} has Status INFORMATION		<code>last(/TPL Planit/status[{#NAME}]) = {#PRIORITY.INFORMATION}</code>	Yes	Yes
<input type="checkbox"/>	Warning	planit Service: {#DISPLAYNAME} has Status WARNING		<code>last(/TPL Planit/status[{#NAME}]) = {#PRIORITY.WARNING}</code>	Yes	Yes

The prototype definitions will automatically create specific triggers, depending on the value of the LLD macro for a given service.

ZDF – Monitoring with Zabbix



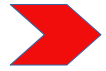
Get states from infrastructure monitoring with dynamic severities using LLD

The result

Planit	Planit Infrastructure Service [message] ?	2021-09-24 18:28:21	System is healthy	Application: Plan.it
Planit	Planit Infrastructure Service [status] ?	2021-09-24 18:28:21	0	Application: Plan.it
Planit	Primary power generator [message] ?	2021-09-24 18:28:21	Generator is healthy.	Application: Powersup...
Planit	Primary power generator [status] ?	2021-09-24 18:28:21	0	Application: Powersup...

Time ▼	<input type="checkbox"/>	Severity	Recovery time	Status	Info	Host	Problem	Duration	Ack	Actions	Tags
18:48:21	<input type="checkbox"/>	Information		PROBLEM		Planit	planit Service: Elasticsearch Service has Status INFORMATION	1m 6s	No		Application: Plan.it
18:48:21	<input type="checkbox"/>	Average		PROBLEM		Planit	planit Service: Transmitting station #3 has Status AVERAGE	1m 6s	Yes	1 →	Application: Radio
18:48:21	<input type="checkbox"/>	Average		PROBLEM		Planit	planit Service: Secondary power generator has Status AVERAGE	1m 6s	No		Application: Powersup...

ZDF – Monitoring with Zabbix



Get states from infrastructure monitoring with dynamic severities using LLD

Well it works, but we can do better

ZDF – Monitoring with Zabbix



Get states from infrastructure monitoring with dynamic severities using LLD

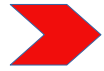
Current solution creates unnecessary triggers for individual components ...

<input type="checkbox"/>	Severity	Value	Name ▲	Operational data	Expression
<input type="checkbox"/>	Average	OK	Planit Discovery: planit Service: Primary power generator has Status AVERAGE		<code>last(/Planit/status[generator-primary]) = -1</code>
<input type="checkbox"/>	Disaster	OK	Planit Discovery: planit Service: Primary power generator has Status DISASTER		<code>last(/Planit/status[generator-primary]) = 2</code>
<input type="checkbox"/>	High	OK	Planit Discovery: planit Service: Primary power generator has Status HIGH		<code>last(/Planit/status[generator-primary]) = 1</code>
<input type="checkbox"/>	Information	OK	Planit Discovery: planit Service: Primary power generator has Status INFORMATION		<code>last(/Planit/status[generator-primary]) = -1</code>
<input type="checkbox"/>	Warning	OK	Planit Discovery: planit Service: Primary power generator has Status WARNING		<code>last(/Planit/status[generator-primary]) = -1</code>

<input type="checkbox"/>	Severity	Value	Name ▲	Operational data	Expression
<input type="checkbox"/>	Average	OK	Planit Discovery: planit Service: Planit Infrastructure Service has Status AVERAGE		<code>last(/Planit/status[planit]) = -1</code>
<input type="checkbox"/>	Disaster	OK	Planit Discovery: planit Service: Planit Infrastructure Service has Status DISASTER		<code>last(/Planit/status[planit]) = -1</code>
<input type="checkbox"/>	High	OK	Planit Discovery: planit Service: Planit Infrastructure Service has Status HIGH		<code>last(/Planit/status[planit]) = 2</code>
<input type="checkbox"/>	Information	OK	Planit Discovery: planit Service: Planit Infrastructure Service has Status INFORMATION		<code>last(/Planit/status[planit]) = -1</code>
<input type="checkbox"/>	Warning	OK	Planit Discovery: planit Service: Planit Infrastructure Service has Status WARNING		<code>last(/Planit/status[planit]) = 1</code>



ZDF – Monitoring with Zabbix



Get states from infrastructure monitoring with dynamic severities using LLD

LLD-Overrides for the rescue

Discovery rule	Preprocessing	LLD macros	Filters	Overrides 5
Overrides				
		Name	Stop processing	Action
		1: PRIORITY.INFORMATION	No	Remove
		2: PRIORITY.WARNING	No	Remove
		3: PRIORITY.AVERAGE	No	Remove
		4: PRIORITY.HIGH	No	Remove
		5: PRIORITY.DISTASTER	No	Remove
		Add		

LLD Overrides were introduced in Zabbix Version 5.0.

ZDF – Monitoring with Zabbix



Get states from infrastructure monitoring with dynamic severities using LLD

LLD-Override configuration – Disable discovery depending on LLD data

The screenshot shows the 'Override' configuration window in Zabbix. The 'Name' field is set to 'PRIORITY.INFORMATION'. Under 'If filter matches', there are two buttons: 'Continue overrides' and 'Stop processing'. In the 'Filters' section, a filter is added with 'Label Macro' as '{#PRIORITY.INFORMATION}', 'Action' as 'matches', and 'Regular expression' as '-1'. In the 'Operations' section, a condition is added: 'Trigger prototype contains INFORMATION'.

Filter: Based on
LLD macro content = -1

Condition: Based on
Trigger prototype name

The screenshot shows the 'Edit operation' window. The 'Object' is 'Trigger prototype'. The 'Condition' is 'contains INFORMATION'. Under 'Create enabled', the 'Discover' checkbox is checked, and the 'Yes' button is highlighted with a red box. There are also 'Original', 'Severity', and 'Tags' options, each with an 'Original' checkbox. At the bottom are 'Update' and 'Cancel' buttons.

```
{
  ...
  "{#PRIORITY.INFORMATION}": -1,
  ...
}
```



ZABBIX
PREMIUM PARTNER

ZDF – Monitoring with Zabbix



Get states from infrastructure monitoring with dynamic severities using LLD

Final result with LLD-Overrides

<input type="checkbox"/>	Severity	Value	Name ▲	Operational data	Expression
<input type="checkbox"/>	Disaster	OK	Planit Discovery: planit Service: Primary power generator has Status DISASTER		<code>last(/Planit LLD/status[generator-primary]) = 2</code>
<input type="checkbox"/>	High	OK	Planit Discovery: planit Service: Primary power generator has Status HIGH		<code>last(/Planit LLD/status[generator-primary]) = 1</code>

<input type="checkbox"/>	Severity	Value	Name ▲	Operational data	Expression
<input type="checkbox"/>	High	OK	Planit Discovery: planit Service: Planit Infrastructure Service has Status HIGH		<code>last(/Planit LLD/status[planit]) = 2</code>
<input type="checkbox"/>	Warning	OK	Planit Discovery: planit Service: Planit Infrastructure Service has Status WARNING		<code>last(/Planit LLD/status[planit]) = 1</code>

ZDF – Monitoring with Zabbix



„Sphinx“ application monitoring using Graylog REST API

Goal

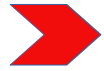
- Use Zabbix for evaluating error messages from the „Sphinx“ application in Graylog (log management) and alert them. Graylog is used for log management only, not for alerting.
- Monitoring the number of errors in user-defined time intervals for different components and alert when a threshold is exceeded.
- Analyse incoming error messages and prepare them for a user friendly output sorted by error types.

Challenges

- How to get the information from Graylog about the Sphinx components (App, Web and WCF Gateway)?
- How to handle certificate problems (DH_KEY_TOO_SMALL / Diffie–Hellman key) due to an outdated version of the installed Graylog server?
- How to sort the error messages coming in „free form“ without explicit error types?



ZDF – Monitoring with Zabbix



„Sphinx“ application monitoring using Graylog REST API

Approach

- Use Zabbix „external check“ item type to solve the certificate problem.
- Configure three master items to make the HTTP API Get request and retrieve the raw data for each component. All additional information is retrieved via dependent item with preprocessing to save further API requests and to be resource friendly.
- Use Zabbix dependent item with Java-Script preprocessing to parse incoming error messages and sort them by error type. Use dependent item for LLD to create the items for the stats data and also the data for the visualization for each error type.
- Create an user friendly dashboard.



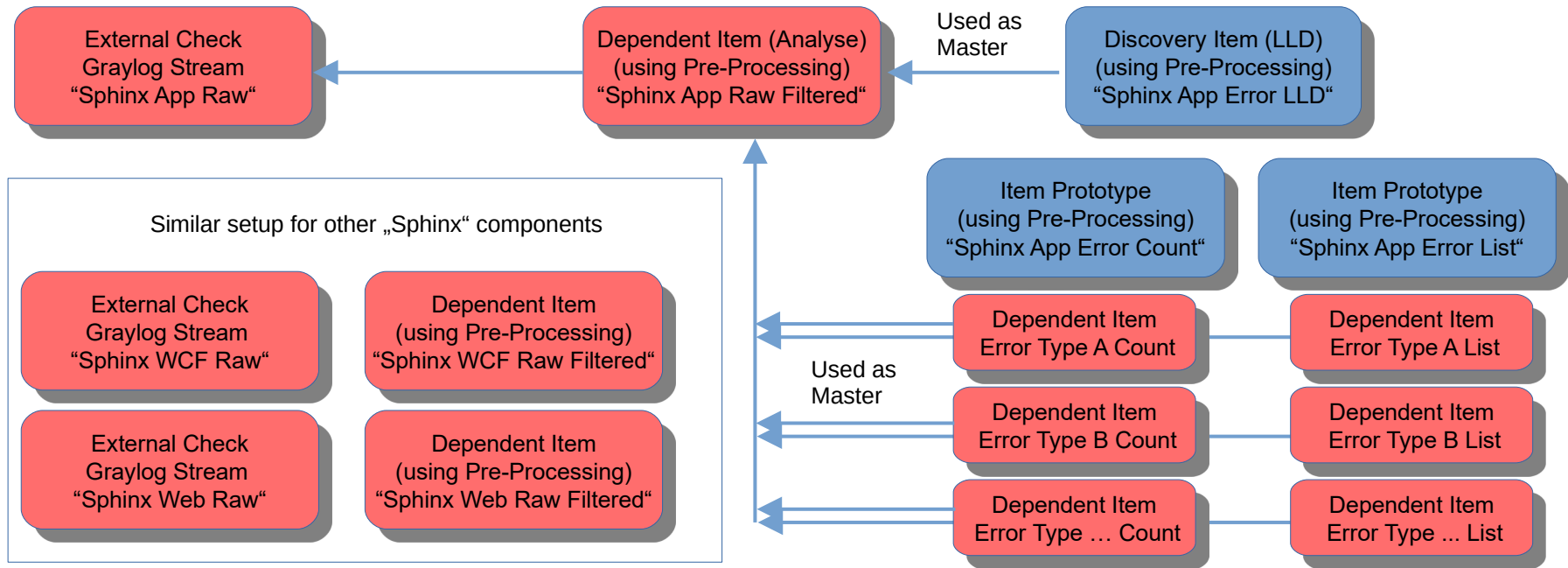
ZABBIX
PREMIUM PARTNER

ZDF – Monitoring with Zabbix

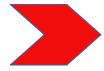


„Sphinx“ application monitoring using Graylog REST API

Item design and dependency using „Sphinx App“ component as an example



ZDF – Monitoring with Zabbix



„Sphinx“ application monitoring using Graylog REST API

Three master items to make the HTTP Get request and retrieve the data for each application level with defined intervals

```
graylog2zabbix.sh[{$GRAYLOG_USERNAME},{ $GRAYLOG_PASSWORD},{HOST.CONN},{ $GRAYLOG_PORT},search/universal/relative?
query=name%3Asphinx-app%20AND%20stage%3Aproduction%20AND%20level%3A(ERROR%20OR
%20FATAL)&range=1800&limit=50&filter=streams%3A60000a8c1c09f9862279966e&fields=name%2Clevel
%2Cmessage&decorate=true]
```

<input type="checkbox"/> Wizard	Name ▾	Triggers	Key	Interval	History	Trends	Type	Applications	Status	Info
<input type="checkbox"/>	Graylog stream Sphinx [sphinx-web] [error / fatal / 60m / RAW]		graylog2zabbix.sh[{\$GRAYLOG_USERNAME},{ \$GRAYLOG_PASSWORD},{HOST.CONN},{ \$GRAYLOG_PORT},search/universal/relative?query=name%3Asphinx-web%20AND%20stage%3Aproduction%20AND%20level%3A(ERROR%20OR%20FATAL)&range=3600&limit=50&filter=streams%3A60000a8c1c09f9862279966e&fields=name%2Clevel%2Cmessage&decorate=true]	5m	1d		External check	raw, sphinx, web	Enabled	
<input type="checkbox"/>	Graylog stream Sphinx [sphinx-wcf] [error / fatal / 120m / RAW]		graylog2zabbix.sh[{\$GRAYLOG_USERNAME},{ \$GRAYLOG_PASSWORD},{HOST.CONN},{ \$GRAYLOG_PORT},search/universal/relative?query=name%3Asphinx-wcf%20AND%20stage%3Aproduction%20AND%20level%3A(ERROR%20OR%20FATAL)&range=7200&limit=50&filter=streams%3A60000a8c1c09f9862279966e&fields=name%2Clevel%2Cmessage&decorate=true]	5m	1d		External check	raw, sphinx, wcf	Enabled	
<input type="checkbox"/>	Graylog stream Sphinx [sphinx-app] [error / fatal / 30m / RAW]		graylog2zabbix.sh[{\$GRAYLOG_USERNAME},{ \$GRAYLOG_PASSWORD},{HOST.CONN},{ \$GRAYLOG_PORT},search/universal/relative?query=name%3Asphinx-app%20AND%20stage%3Aproduction%20AND%20level%3A(ERROR%20OR%20FATAL)&range=1800&limit=50&filter=streams%3A60000a8c1c09f9862279966e&fields=name%2Clevel%2Cmessage&decorate=true]	5m	1d		External check	app, raw, sphinx	Enabled	

Displaying 3 of 3 found

Application level: App

Application level: WCF

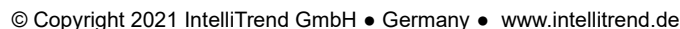
Application level: Web



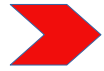
ZABBIX
PREMIUM PARTNER

„Sphinx“ application monitoring using Graylog REST API

ZABBIX
PREMIUM PARTNER



ZDF – Monitoring with Zabbix



„Sphinx“ application monitoring using Graylog REST API

Zabbix dependent item to analyse the error messages

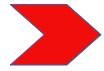
The screenshot shows the Zabbix web interface for configuring a new item. The breadcrumb trail is: All templates / ZDF Template Graylog [Sphinx] / Applications 9 / Items 12. The 'Preprocessing' tab is active. The configuration fields are as follows:

- Name:** Graylog stream Sphinx [sphinx-web] [error / fatal / 60m] [Filtered]
- Type:** Dependent item
- Key:** sphinxWebMessagesFiltered
- Master item:** ZDF Template Graylog [Sphinx]: Graylog stream Sphinx [sphinx-web] [error / ...]
- Type of information:** Text
- History storage period:** Do not keep history (selected) / Storage period: 90d
- New application:** (empty text box)
- Applications:** -None-app

Pre-Processing Steps:

- Analyse the error messages.
- Define the error type.
- Sort the raw data.

ZDF – Monitoring with Zabbix



„Sphinx“ application monitoring using Graylog REST API

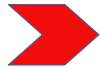
Preprocessing steps

All templates / ZDF Template Graylog [Sphinx] Applications 9 Items 12 Triggers 6 Graphs 1 Screens 1 Discovery rules 3 Web scenarios

Item Preprocessing

Preprocessing steps	Name	Parameters	Custom on fail	Actions
1:	JSONPath	<input type="text" value="\$messages[*].message.message"/>	<input checked="" type="checkbox"/>	Test Remove
	Custom on fail	<input type="text" value="Discard value"/> <input type="text" value="Set value to"/> <input type="text" value="Set error to"/> <input type="text" value="{}"/>		
2:	JavaScript	<input type="text" value="var errorList = JSON.parse(value);..."/>	<input type="checkbox"/>	Test Remove
Add				Test all steps

ZDF – Monitoring with Zabbix



„Sphinx“ application monitoring using Graylog REST API

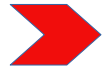
Java script to define the error message type and sort the messages

```
JavaScript
function (value) {
  1 var errorList = JSON.parse(value);
  2
  3 // get unique list of error types
  4 //^(.*?)[\.\!|:|][\s|\r|\n]
  5 var regex = new RegExp(/^(.*?)[\.\!|:|][\s|\r|\n]/);
  6 var result = {};
  7 for (var i = 0; i < errorList.length; i++) {
  8   var errorType = regex.exec(errorList[i]);
  9   if (errorType == null) {
 10     errorType = "Undefined";
 11   } else {
 12     errorType = errorType[1];
 13   }
 14   // add error to result array if existing or create new array in dict
 15   if (errorType in result) {
 16     result[errorType].push(errorList[i]);
 17   } else {
 18     result[errorType] = [errorList[i]];
 19   }
 20 }
 21 return JSON.stringify(result);
}
```

64920 symbols remaining

Apply Cancel

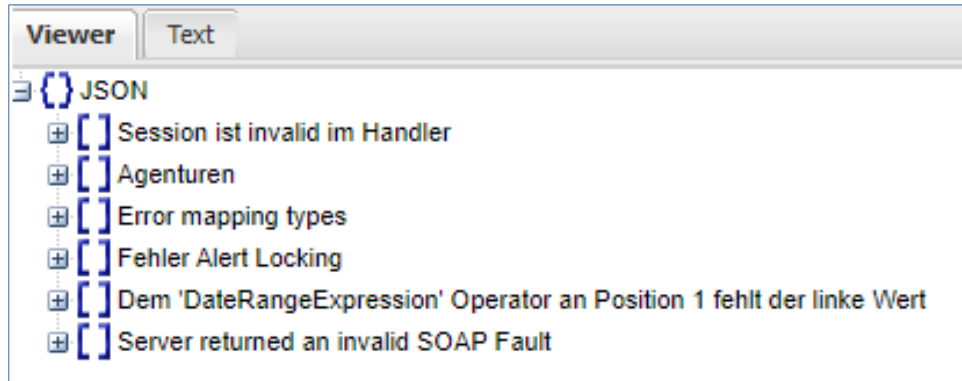
ZDF – Monitoring with Zabbix



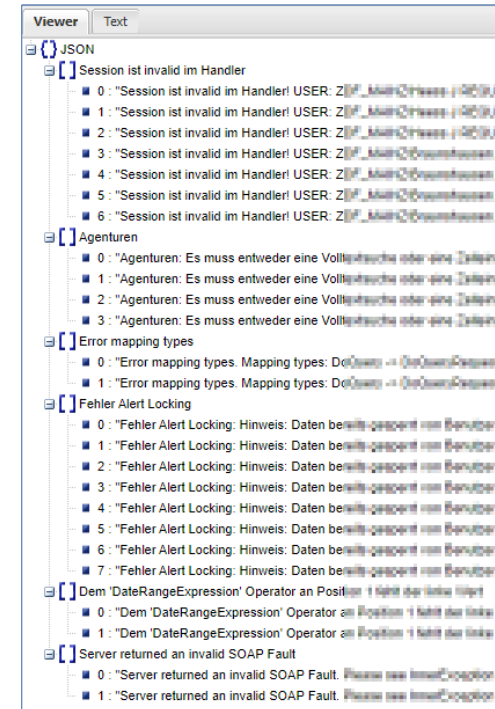
„Sphinx“ application monitoring using Graylog REST API

Error messages after preprocessing sorted by type

Error types

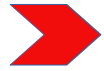


Error types with errors



ZABBIX
PREMIUM PARTNER

ZDF – Monitoring with Zabbix



„Sphinx“ application monitoring using Graylog REST API

Dependent LLD item to define the error message type and sort the raw data

<input type="checkbox"/> Host	Name ▲	Items	Triggers	Graphs	Hosts	Key	Interval	Type	Status	Info
<input type="checkbox"/> ZDF Template Graylog [Sphinx]	Graylog stream Sphinx [sphinx-app] [error / fatal / 30m] [Filtered]: Sphinx app backend error parser LLD	Item prototypes 2	Trigger prototypes 1	Graph prototypes	Host prototypes	sphinx.app.error.lld		Dependent item	Enabled	
<input type="checkbox"/> ZDF Template Graylog [Sphinx]	Graylog stream Sphinx [sphinx-wcf] [error / fatal / 120m] [Filtered]: Sphinx wcf gateway error parser LLD	Item prototypes 2	Trigger prototypes 1	Graph prototypes	Host prototypes	sphinx.wcf.error.lld		Dependent item	Enabled	
<input type="checkbox"/> ZDF Template Graylog [Sphinx]	Graylog stream Sphinx [sphinx-web] [error / fatal / 60m] [Filtered]: Sphinx web frontend error parser LLD	Item prototypes 2	Trigger prototypes 1	Graph prototypes	Host prototypes	sphinx.web.error.lld		Dependent item	Enabled	

Displaying 3 of 3 found

Discovery rules

All templates / ZDF Template Graylog [Sphinx] / Discovery list / Sphinx web frontend error parser L... / Item prototypes 2

Discovery rule Preprocessing LLD macros Filters Overrides

Name **Sphinx web frontend error parser LLD**

Type **Dependent item**

Key **sphinx.web.error.lld**

Master item **ZDF Template Graylog [Sphinx]: Graylog stream Sphinx [sphinx-web] [error /**

Keep lost resources period **30d**

Description **Create an item for each error type from sphinx frontend (web) log**

Enabled ☒

Update **Clone** **Test** **Delete** **Cancel**

All templates / ZDF Template Graylog [Sphinx] / Discovery list / Sphinx web frontend error parser L... / Item prototypes 2 / Trigger prototypes 1 / Graph prototypes / Host proto

Discovery rule Preprocessing LLD macros Filters Overrides

Preprocessing steps

Name	Parameters
1 JSONPath	[\$]~
2 JavaScript	var errorTypes = JSON.parse(value);...

Add **Update** **Clone** **Test** **Delete** **Cancel**

JavaScript

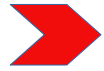
```
function (value) {  
  1 var errorTypes = JSON.parse(value);  
  2 var result = {};  
  3  
  4 for (var key in errorTypes) {  
  5   result.push({  
  6     "ERROR.TYPE": errorTypes[key]  
  7   });  
  8 }  
  9  
 10 return JSON.stringify(result);  
}
```

65351 symbols remaining

Apply **Cancel**

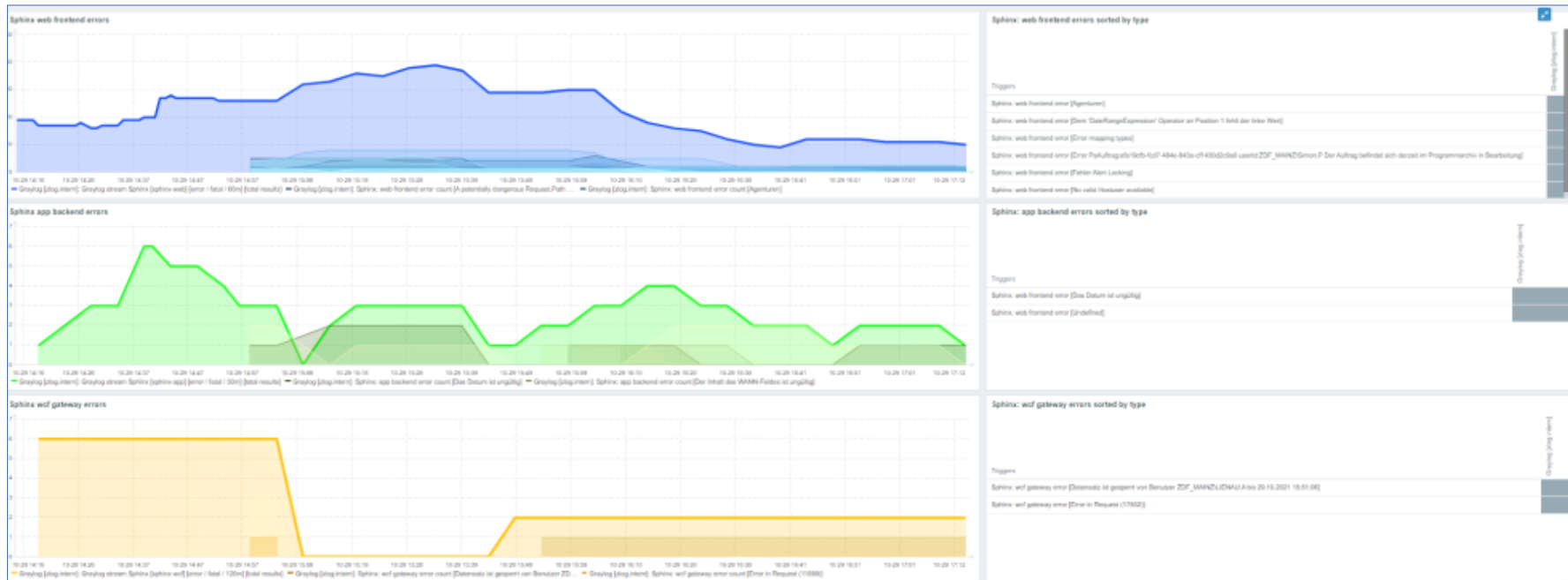


ZDF – Monitoring with Zabbix



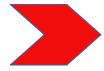
„Sphinx“ application monitoring using Graylog REST API

Everyone loves dashboard



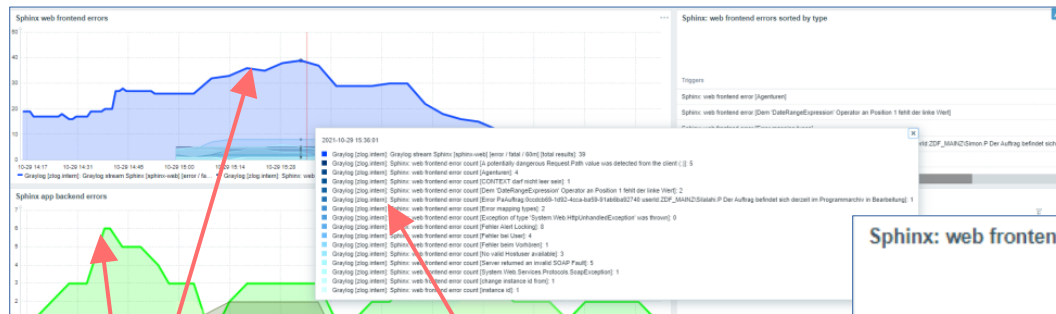
ZABBIX
PREMIUM PARTNER

ZDF – Monitoring with Zabbix



„Sphinx“ application monitoring using Graylog REST API

Everyone loves dashboard



Total number of errors per component

Number of errors per error type

List of active error types per component

Sphinx: web frontend errors sorted by type	
Triggers	
Sphinx: web frontend error [Agenturen]	
Sphinx: web frontend error [Dem 'DateRangeExpression' Operator an Position 1 fehlt der linke Wert]	
Sphinx: web frontend error [Error mapping types]	
Sphinx: web frontend error [Error PaAuftrag:efa19cfb-fcd7-484e-b43a-cf1436c2c2a0 userid:ZDF_MAIKZ:Simon P Der Auftrag befindet sich der	
Sphinx: web frontend error [Fehler Alert Locking]	
Sphinx: web frontend error [No valid Hostuser available]	



ZABBIX
PREMIUM PARTNER

ZDF – Monitoring with Zabbix

Monitor something different

ZDF – Monitoring with Zabbix



TV broadcast truck („Übertragungs-Wagen“)

ZDF – Monitoring with Zabbix



Monitoring a TV broadcast truck („Übertragungs-Wagen“)

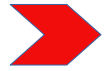
Goal

- Monitor several metrics from different technologies used in the TV broadcast truck.
- Monitor communication availability and quality between the broadcast truck and the transmitting station.
- Only monitor the broadcast truck when in use.

Challenges

- How can false positive alarms be avoided if a broadcast truck can be put into operation spontaneously (without notifying the monitoring team)?

ZDF – Monitoring with Zabbix



Monitoring a TV broadcast truck („Übertragungs-Wagen“)

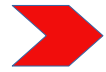
Approach

- Treat a broadcast truck and its components as a host that can be put into maintenance.
- Create a control host (as entity in Zabbix) to monitor the connection states of all broadcasting trucks.
- Create a middleware that implements a smart logic to start/stop monitoring a given broadcasting truck by switching maintenance using the Zabbix API.
- A specific application in the broadcasting truck then tells Zabbix how long to monitor and when to enable maintenance again.

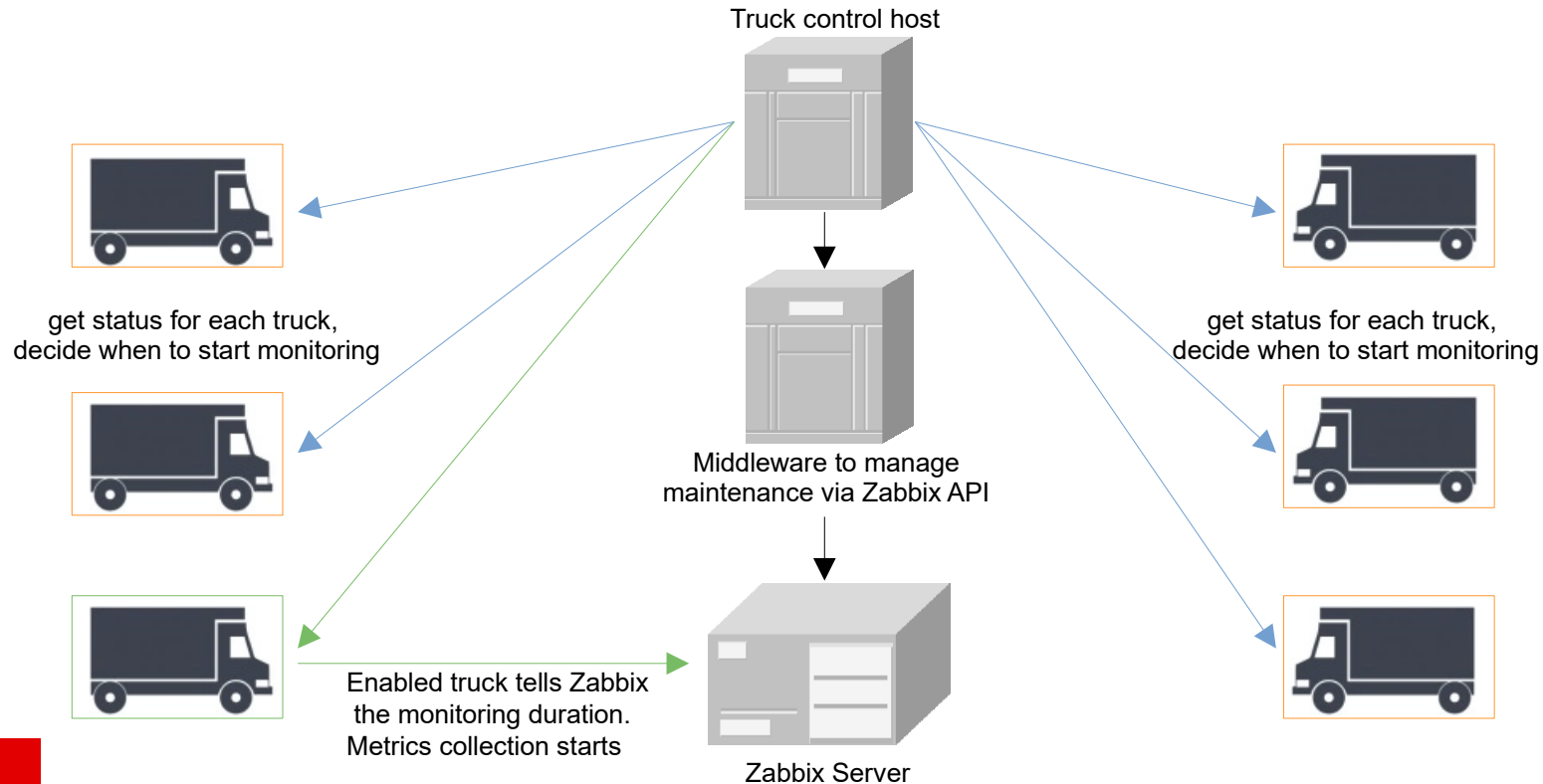


ZABBIX
PREMIUM PARTNER

ZDF – Monitoring with Zabbix



Monitoring a TV broadcast truck („Übertragungs-Wagen“)

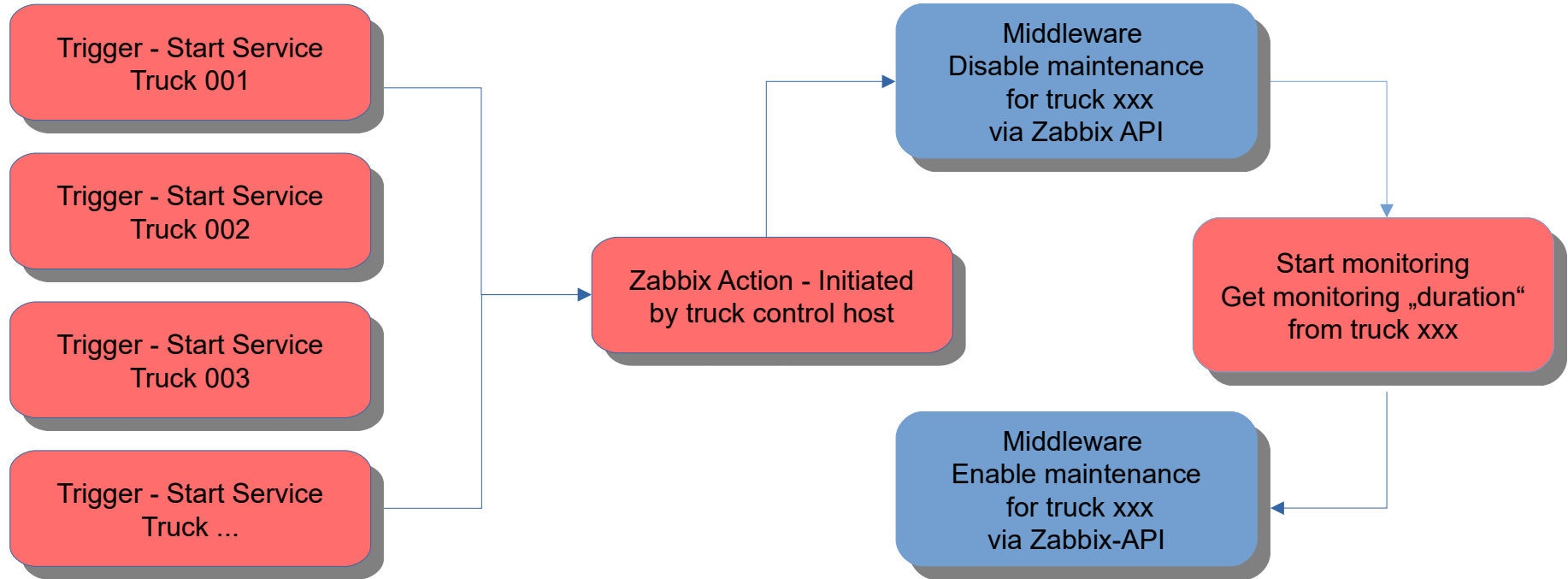


ZABBIX
PREMIUM PARTNER

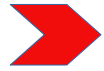
ZDF – Monitoring with Zabbix



Monitoring a TV broadcast truck („Übertragungs-Wagen“)



ZDF – Monitoring with Zabbix



Monitoring a TV broadcast truck („Übertragungs-Wagen“)

The screenshot shows the Zabbix Monitoring interface. The sidebar on the left contains the following menu items: Monitoring (selected), Dashboard, Problems, Hosts, Overview, Latest data, Screens, Maps, Discovery, Services, Inventory, Reports, Configuration, Administration, Support, Share, Help, User settings, and Sign out. The main area displays a grid of 25 TV broadcast trucks, each represented by a truck icon and a label. The labels are as follows:

Uewagen ID	Status
Uewagen-001	In maintenance OK
Uewagen-002	In maintenance OK
Uewagen-003	OK
Uewagen-004	In maintenance OK
Uewagen-005	In maintenance OK
Uewagen-006	In maintenance OK
Uewagen-007	In maintenance OK
Uewagen-008	In maintenance OK
Uewagen-009	OK
Uewagen-010	OK
Uewagen-011	In maintenance OK
Uewagen-012	In maintenance OK
Uewagen-013	In maintenance OK
Uewagen-014	In maintenance OK
Uewagen-015	In maintenance OK
Uewagen-016	OK
Uewagen-017	Keine Verbindung zur Sendeanstalt
Uewagen-018	In maintenance OK
Uewagen-019	In maintenance OK
Uewagen-020	In maintenance OK
Uewagen-021	OK
Uewagen-022	In maintenance OK
Uewagen-023	In maintenance OK
Uewagen-024	In maintenance OK
Uewagen-025	Keine Verbindung zur Sendeanstalt
Uewagen-026	In maintenance OK
Uewagen-027	In maintenance OK
Uewagen-028	In maintenance OK
Uewagen-029	In maintenance OK
Uewagen-030	In maintenance OK



ZABBIX
PREMIUM PARTNER

Zabbix meets television

Clever use of Zabbix features



IntelliTrend GmbH

www.intellitrend.de

Thank You!



Contact: Wolfgang Alper

wolfgang.alper@intellitrend.de

