Central Monitoring Service

This is a copy of the original document as a work example.
All links to internal documents have been removed.

Central Monitoring - Most Important Problem	

THE PROBLEM	We are failing to detect or prevent outages because our monitoring **quality bar and coverage** is not good enough:
	- **quality bar**
	  - we do not have a quality bar to define what "good monitoring" is for a product, instead we are implementing monitors reactively, when PCAs demand it or as a reaction to an outage.
	  - we do not monitor the availability or performance of product services, we just monitor if the infrastructure is running.
	  - we use a complex, self-managed Zabbix monitoring solution which is not a good fit for monitoring native AWS services or dynamically instantiated K8 pods / Docker containers
	- **coverage**
	  - we only monitor a third of the products running in Central Engineering (33 SE7 units from this list)
	  - the rest (29 SE7 SaaS units and 40 SE7 onprem units) are not monitored and we're not including them in our uptime reporting. The monitoring for these products, if any, is not standardized and not centrally maintained).

RELATED DEEP DIVES	Kayako - performance issues Jive - missing LDAP disk monitor Jive - missing SQS monitor StreetSmart - missing DB monitor StreetSmart - missing service monitor Lyris HQ - missing DB connectivity monitor Lyris HQ - missing log monitor Lyris HQ - missing Docker monitor Manuscript - missing service monitor Manuscript - missing IIS monitor Manuscript - missing DB monitor AlertFind - compromised synthetic monitoring credentials led to security incident EMS - inaction of monitoring alerts led to outage CRM - missing service monitor SLI - lift and shifting monitoring Important finding: we have 1500 docker containers and 1600 K8s pods running in our Central Docker and K8. We are not @monitoring any of them because we previously assumed they are part of Central Platform monitoring scope. This assumption is incorrect, Central Platform is only monitoring the Central hosts the clusters are running on, not the containers and pods *within* the clusters.
INSIGHTS FROM DEEP DIVES	***Insight: our infrastructure monitoring is incomplete, we are missing important monitors.*** - example: see "missing LDAP/SQS/DB/IIS monitor" deep dives above. ***Insight: once we fix the infrastructure monitoring, we also need to do application monitoring to prevent/detect outages caused by product services being down.*** - customers experience outages not only when the infrastructure is down, but also when the product performance is heavily degraded: pages load slowly or intermittently fail to load. - such issues may be caused by product services being down or partially affected by an internal issue that is unrelated to the underlying infrastructure. - today we are failing to prevent or immediately detect such outages because our monitoring only checks if the infrastructure is up, but it does not do any kind of product service/application monitoring. - example: see "Kayako - performance issues" and other "missing service monitor" deep dives above. ***Insight: our current monitoring solution (self-managed Zabbix) is not a good fit for AWS*** - Zabbix is not designed to monitor AWS services and dynamic fleets of containers/pods. We need to build and maintain custom scripts to get the level of monitoring that comes with CloudWatch out of the box.
SOLUTION SUMMARY	We will fix the monitoring **quality bar** by: - monitoring both the infrastructure and the product services running on top of the infrastructure; we will use our core DSs (SaaS DS, C4 DS) to identify what is important to be monitored (ITD.MONITOR.1). - replacing our monitoring framework; we will move from Zabbix to AWS CloudWatch (ITD.MONITOR.2). - standardizing the metrics we are monitoring; we will reuse the monitor implementation across products (ITD.MONITOR.3).

	We will fix the monitoring **coverage** by: - importing all SaaS products we are currently not monitoring using this framework. - once the new imports are complete, we will re-import the other SaaS products too.
PROOF OF CONCEPT	To prove our solution, we took one product - AlertFind - and: - we have documented the current state of its monitoring. - we have implemented the new state of its monitoring by applying the ITDs in this document.

ITD.MONITOR.1 - Implement Infrastructure and Application monitoring	
THE PROBLEM	We need to choose the type of monitoring that would detect or prevent all Central SaaS outages.
OPTIONS CONSIDERED	Option 1 - Do Infrastructure monitoring only. Current state. **Option 2 - Do Infrastructure and Application monitoring.**
REASONING	*Option 1 - Do Infrastructure monitoring only. Current state.* We are not choosing this option because a product may still experience an outage even if the infrastructure it is running on is up and running. Evidence: Kayako performance issues. ***Option 2 - Do Infrastructure and Application monitoring.*** We are choosing this option because we want to determine if the infrastructure is up and running and the product is operating as expected on top of this infrastructure. This is what we consider is important to monitor and why: 1. For the **Infrastructure monitoring** we will monitor if all infrastructure elements are running and utilized below 90% capacity: - it is important to monitor if "**infrastructure elements are running**" because we assume all infrastructure needs to be operational for the product to work correctly. - it is important to monitor if "**infrastructure elements are utilized below 90% capacity**" because infrastructure capacity exhaustion, be it CPU, memory, storage or bandwidth is our second most frequent outage cause after the infrastructure being down. 2. For the **Application monitoring** we will monitor if all product services are running and if the products's external interfaces (APIs) are responding without errors, with an acceptable level of performance (industry-standard: <3s response time): - it is important to monitor if "**services are running**" because we assume all services need to be operational for the product to work correctly. - it is important to monitor if "**APIs are responding without errors**" because we believe any HTTP response code in the 4/500 range is indicating an unexpected error that requires our attention because it may be the symptom of a product defect or an outage. - it is important to monitor if "**APIs are responding with an acceptable level of performance**" because extreme slowness is perceived as an outage by customers and is a strong indicator of a problem that will rapidly turn into a full-service-down outage. This matrix summarizes our monitoring framework:

MONITORING SCOPE		**IMPLEMENTATION** *As per ITD.MONITOR.2*	**QUALITY BAR** *This is the definition of "good monitoring" at product level*	**ACTION IF CONDITION BREACHED**
1. Infrastructure	1.1. Availability	CloudWatch	All infrastructure elements (identified from SaaS DS) are running?	Trigger CloudWatch Alarms so SaaS Support L1 can execute the runbook associated with the specific alarm event.
	1.2. Utilization		All infrastructure elements are utilized below 90% of their available capacity?	
2. Applications	2.1. Availability	CloudWatch Agents	All products= services (identified from C4 DS) are running?	
	2.2. Functionality	CloudWatch Synthetic Canaries	All calls to external application interfaces (APIs, identified from C4 DS) are not returning 4/5xx error codes?	
	2.3. Performance		All calls external application interfaces are returning in under 3 seconds (default threshold)	

ITD.MONITOR.2 - Implement monitoring with AWS CloudWatch	
THE PROBLEM	We need to define the implementation pattern (monitoring tool) for our monitoring framework.
OPTIONS CONSIDERED	Option 1 - Zabbix. Current state. **Option 2 - CloudWatch** Option 3 - Prometheus Option 4 - CloudWatch for AWS services and Prometheus for everything else

REASONING	Option 1 - *Zabbix. Current state. We are not choosing this option because:*
	- Zabbix is self-managed, thus adding complexity related to deploying, maintaining and monitoring Zabbix servers.
	- Zabbix is a static monitoring tool, not a good fit to monitor infrastructure elements that are created and torn down dynamically - such as Docker containers, Kubernetes pods, Spot instances or Autoscaling groups:
	  - Zabbix requires a client agent to be installed on all monitored instances (containers, pods, hosts).
	  - When new instances are created, the Zabbix agent needs to be registered with the Zabbix server, this does not happen automatically.
	  - Thus we have to create scripts to register each new agent with the Zabbix server.
	  - These scripts are adding complexity: they need to be maintained and updated every time there is a product architecture change.
	
	Option 2 - *CloudWatch.* We are choosing this option because:
	- CloudWatch is the default monitoring service provided by AWS, thus removing the self-manage overhead associated with other monitoring tools.
	- CloudWatch provides out-of-the-box monitors for all attributes we want to monitor as per ITD.MONITOR.1
	  - Infrastructure availability and utilization - CloudWatch
	  - Product availability - CloudWatch Agents
	  - Product functionality and performance - CloudWatch Synthetic Canaries
	- CloudWatch provides machine-learning algorithms to detect anomalies - we will enable CloudWatch Anomaly Detection
	- CloudWatch is a great fit for dynamic infrastructures: the CloudWatch agent can be deployed as a DaemonSet to collect cluster metrics and automatically start collecting EKS container insights.
	
	Option 3 - *Prometheus.* We are not choosing this option because:
	- Prometheus, just like Zabbix, it cannot natively monitor AWS services. Prometheus requires a client library to be deployed on the target resource, which is not possible with AWS services.
	- The workaround limitation would be to set up a host inside the AWS account which monitors the AWS service and feeds the metrics to a Prometheus exporter. This is an unnecessarily complex architecture considering CloudWatch can monitor these services natively.
	
	Option 4 - *CloudWatch for AWS services and Prometheus for everything else.* We are not choosing this option because CloudWatch offers all of the capabilities that we need from Prometheus in addition to AWS service monitoring so there is no need to proliferate monitoring solutions.

ITD.MONITOR.3 - Define default CloudWatch monitoring metrics for each type of infrastructure element and product service.	
THE PROBLEM	We need to define which CloudWatch monitoring metrics are most relevant for monitoring: - the infrastructure availability and utilization - the product availability, functionality and performance.
OPTIONS CONSIDERED	Option 1 - SaaS PKCs make a product-level decision what CloudWatch metrics to use. Option 2 - Enable all CloudWatch metrics. **Option 3 - Define default monitor to use for each type of infrastructure element and product service.**
REASONING	Option 1 - *SaaS PKCs make a product-level decision what CloudWatch monitors to use*. We are not choosing this option because we do not want to make one-off decisions. Our products run on similar infrastructures and use similar services, so standardizing how we do monitoring is a better option than making product-level decisions. Option 2 - *Enable all CloudWatch monitors*. We are not choosing this option because it would be excessive, creating more noise than signal. Our goal is to detect and prevent outages and to achieve this goal we do not need to look at the hundreds of metrics exposed by AWS CloudWatch. **Option 3 - *Define default CloudWatch monitoring metrics for each type of infrastructure element and product service***. We

are choosing this option because we want to standardize monitoring. We created this set of default monitoring decisions for all the infrastructure elements and product services currently used by our products:

Infrastructure Elements - Monitoring Decisions			
INFRASTRUCTURE CLASS	**INFRASTRUCTURE ELEMENTS**	**IMPLEMENTATION**	**DEFAULT METRIC AND THRESHOLD**
Compute	API Gateway	CloudWatch	4XXError > 2% for 15 min (runbook) 5XXError > 2% for 15 min (runbook) Latency > 3s for 5 min (runbook)
	Autoscaling Group	CloudWatch	DesiredVsInService >= 1 for 5 min (runbook)
	Cloudfront	CloudWatch	Total error rate > 10% for 5 min (runbook)
	EC2 Hosts	CloudWatch	Availability = false *(runbook)* CPUUtilization > 99% for 5 minutes (runbook) StatusCheckFailed_Instance = 1 for 5 min (runbook) StatusCheckFailed_System = 1 for 5 min (runbook)
		CloudWatch Agent (Linux)	disk_used_percent > 90% for 5 min (runbook) Mem_used_percent > 95% for 10 min (runbook)
		CloudWatch Agent (Windows)	LogicalDisk % Free Space < 10% for 5 min (runbook) AWS Health Events Monitor *(WIP)*
	ECS	CloudWatch	Availability = false CpuUtilized = 100% for 5 minutes (runbook) CpuReserved = 100% for 5 minutes MemoryUtilized > 2σ over past week for 5 min *(WIP)* MemoryReserved > 2σ over past week for 5 min *(WIP)* PendingTaskCount > 0 for 5 min *(WIP)*
		CloudWatch Agent	Instance_cpu_utilization = 100% for 5 minutes *(WIP)* Instance_filesystem_utilization > 90% for 5 minutes *(WIP)* Instance_memory_utiliization > 90% for 5 minutes *(WIP)*
	EKS	CloudWatch Agent DaemonSet	Availability = false cluster_failed_node_count > 0 for 5 minutes *(WIP)* Node_filesystem_utilization > 90% for 5 minutes (WIP) node_cpu_utilization > 99% for 5 min *(WIP)* node_memory_reserved_capacity > 90% for 5 min *(WIP)* node_memory_utilization > 90% for 5 min *(WIP)* pod_cpu_utilization > 99% for 5 minutes *(WIP)* pod_cpu_utilization_over_pod_limit > 5% for 5 minutes *(WIP)* pod_memory_utilization > 90% for 5 minutes *(WIP)* pod_memory_utilization_over_pod_limit > 0% for 5 minutes *(WIP)*
	K8		
	ELB	CloudWatch	UnHealthyHostRate > 32% for 5 min (runbook)
	ES	CloudWatch	ClusterStatus.red >= 1 for 1 min (runbook) ClusterStatus.yellow >= 1 for 1 min (runbook) FreeStorageSpace <= 20480 for 1 min *(runbook)* ClusterIndexWritesBlocked >= 1 for 5 min (runbook) AutomatedSnapshotFailure >= 1 for 1 min (runbook) CPUUtilization >= 80% for 15 min (runbook) JVMMemoryPressure >= 80% for 5 min (runbook) MasterCPUUtilization >= 50% for 15 min *(runbook)* MasterJVMMemoryPressure >= 80% for 15 min *(runbook)* KMSKeyError >= 1 for 1 min *(runbook)* KMSKeyInaccessible >= 1 for 1 min *(runbook)*
	Lambda	CloudWatch using Lambda Insights	errors / invocations > 0.1 for 5 min *(runbook)* throttles / invocations > 0.1 for 5 min *(runbook)* Duration > 2σ over past week for 5 min (runbook) IteratorAge > 2σ over past week for 5 min *(runbook)* ProvisionedConcurrencyUtilization > 0.9 for 5 min *(runbook)* Ingress Lambda Concurrencies > 480 for 5 min *(WIP)*
	SES	CloudWatch	Reputation.BounceRate > 0.05 for 1h *(runbook)* Reputation.ComplaintRate > 0.001 for 1 h *(runbook)*

	SNS	CloudWatch	NumberOfNotificationsFailedPercentage >= 5 for 5 min (runbook)
	SQS	CloudWatch	ApproximateAgeOfOldestMessage > 2σ over past week for 5 min (runbook) ApproximateNumberOfMessagesNotVisible > 110,000 (runbook) NumberOfMessagesSent > 2σ over past week for 15 min (runbook)
	VMC Hosts	CloudWatch Agent	Availability = false (*runbook*) Instance_cpu_utilization = 100% for 5 minutes (*runbook)* Instance_filesystem_utilization > 90% for 5 minutes Instance_memory_utiliization > 90% for 5 minutes (*runbook)*
Databases	RDS (MySQL) (incl Aurora)	CloudWatch	Availability = false (runbook) CPUUtilization >= 95% for datapoints during 15 min (runbook) DatabaseConnections > 3σ over past week for 3 datapoints during 15 min (runbook) Free Local Storage < 10% (runbook) FreeableMemory < 5 % (runbook) AuroraBinLogReplicaLag > 3s (runbook**)** AuroraVolumeBytesLeftTotal > 100 TB (Aurora only) (runbook)
	RDS (Oracle)	CloudWatch	Availability = false (runbook) CPUUtilization >= 95% for datapoints during 15 min (runbook) DDatabaseConnections > 3σ over past week for 3 datapoints during 15 min (runbook) DiskQueueDepth > 1000 for 5 minutes (runbook) Free Local Storage < 5% (runbook) FreeableMemory < 5 % (runbook)
	RDS (PostgresQL) (incl Aurora)	CloudWatch	Availability = false (runbook) CPUUtilization >= 95% for datapoints during 15 min (runbook) DatabaseConnections > 3σ over past week for 3 datapoints during 15 min (runbook) DiskQueueDepth > 1000 for 5 minutes (runbook) Free Local Storage < 5% (runbook) FreeableMemory < 5 % (runbook) SwapUsage > 1MB (runbook) RDSToAuroraPostgreSQLReplicaLag > 3s (runbook**)** MaximumUsedTransactionIDs > 1000000000 (runbook)
	Elasticache - Redis	CloudWatch	Availability = false (runbook) CPUUtilization > 90% for 5 min (runbook) EngineCPUUtilization > 90% for 5 min (runbook) SwapUsage > 50MB (runbook) DatabaseMemoryUsagePercentage > 90% for 5 min (runbook)
	DynamoDB	CloudWatch	Availability = false MaxProvisionedTableReadCapacityUtilization > 80% for 5 min (runbook) MaxProvisionedTableWriteCapacityUtilization > 80% for 5 min (runbook) Sustained read throttling > 2% for 5 min Sustained write throttling > 2% for 5 min Sustained significant elevation of system errors > 2% for 5 min Sustained significant elevation of user errors > 2% for 5 min ConditionalCheckFailedRequests > 100 for 5 min TransactionConflict > 100 for 5 min
	Aurora	CloudWatch	Availability = false CPUUtilization = 100% for 5 min FreeLocalStorage < 10GB

	Aurora Cluster	CloudWatch	Availability = false AuroraVolumeBytesLeftTotal > 100 TB (Aurora only) (runbook)
Storage	EBS	CloudWatch	VolumeThroughputPercentage < 90% for 5 min (runbook) BurstBalance < 20% for 5 min (runbook)
	Kinesis	CloudWatch	GetRecords.IteratorAgeMilliseconds > 50% (runbook) GetRecords.Success > 2σ over past week for 5 min *(runbook)* PutRecord.Success > 2σ over past week for 5 min *(runbook)* PutRecords.Success > 2σ over past week for 5 min *(runbook)* ReadProvisionedThroughputExceeded > 2σ over past week for 5 min (runbook) WriteProvisionedThroughputExceeded > 2σ over past week for 5 min (runbook)
	EFS	CloudWatch	Mount Status = "Mount failed" on EC2 instance for 5 min (runbook) Throughput utilization > 90% for 15 min (runbook) Percent IO limit = 100% for 5 min (runbook)
	S3	CloudWatch	
Network	Cloudfront	CloudWatch	Availability = false Total error rate >10% for 5 data points 5 minutes each (runbook)
	VPC	CloudWatch + VPC Reachability Analyzer	Availability = false
	Route53	CloudWatch	ConnectionTime > 500ms (runbook) HealthCheckPercentageHealthy < 100% for 5 min (runbook) HealthCheckStatus = 0 for 5 min (runbook) SSLHandshakeTime > 1s (runbook) TimeToFirstByte > 3s (runbook) DNSSECInternalFailure = 1 for 5 min (runbook) DNSSECKeySigningKeysNeedingAction = 1 for 5 min (runbook)

Product Services - Monitoring Decisions			
SERVICE CLASS	**SERVICE INSTANCE**	**IMPLEMENTATION**	**DEFAULT METRIC AND THRESHOLD**
Application Services	Tomcat	Cloud Watch Agent	Availability = false
	Apache HTTP		
	Nginx		
	Kafka		
	Zookeeper		
	IIS		
	MQ Services		
	Docker		
	cron		
	crond		
	snmpd		
	sendmail		
	MailScanner		
	rsyslog		
	syslog		
	ntp		
	portreserve		
	Collectd		
	Postfix		
	vsftpd		

	proftpd		
	sshguard		
	nagios		
	nfs-kernel-server		
	strongmail		
	graylog		
	Database Migration Service		
	Simple Notification Service		
	OpenVPN		
	Consul		
	Vault		
	Docker Swarm		
	Nomad		
	MSDC		
SSL Certificates	SSL Certificate	Certificate check	Expiration < 5 days

AlertFind

ALERTFIND - “BEFORE” MONITORING

Problems with current state:

- SaaS Support does not respond to centralized services:
  - Central EKS, Central Docker, Central DB, Central Networking
  - Alerts are routed directly to the Central SaaS team who does **not** provide 24x7 support
- Monitors are ad-hoc and product specific, not based on standards and best-practices
- There are huge gaps in our monitoring because we only monitor a small sampling of product functionality instead verifying that the entire app is working properly
- Synthetic monitors are complex and require PCA input to create instead of verifying app functionality through its API

ALERTFIND - INFRASTRUCTURE MONITORS			
INFRASTRUCTURE CLASS	**INFRASTRUCTURE ELEMENTS**	**IMPLEMENTATION**	**DEFAULT METRIC AND THRESHOLD**
Compute	moafcid-laf-provato-bridge	Zabbix	/etc/passwd has been changed Configured max number of opened files is too low on Configured max number of processes is too low on CPU utilization is high CPU utilization is very high Disk I/O is overloaded Free disk space is less than 90% on volume / Free disk space is less than 95% on volume / Free inodes is less than 10% on volume / Free inodes is less than 20% on volume / Host information was changed Host name of zabbix_agentd was changed Hostname was changed Lack of free swap space Memory utilization is high Memory utilization is very high Processor load is too high Too many processes (critical) Too many processes Too many processes running Too many zombie processes

			Version of zabbix_agent(d) was changed Zabbix agent is unreachable for 1 hour Zabbix agent is unreachable for 5 minutes Host has just been restarted

ALERTFIND - APPLICATION MONITORS			
SERVICE CLASS	**SERVICE INSTANCE**	**IMPLEMENTATION**	**DEFAULT METRIC AND THRESHOLD**
Application Service	messageone-prod-caf-messageone-aureacentral-com	Zabbix	AlertFind Scheduler service stuck in database
	messageone-prod-laf-messageone-aureacentral-com		AlertFind Scheduler service stuck in database
	AlertFind Server		messageone-af-s-chi-easier-login-check synthetics failure
			messageone-af-s-chi-email-notifications synthetics failure
			messageone-af-s-chi-login-check synthetics failure
			messageone-af-s-chi-msdc-notifications synthetics failure
			messageone-af-s-demo-login-check synthetics failure
			messageone-af-s-lon-easier-login-check synthetics failure
			messageone-af-s-lon-email-notifications synthetics failure
			messageone-af-s-lon-login-check synthetics failure
			messageone-af-s-msdc-mobile-app-login synthetics failure
			messageone-af-s-ping-msdc-version-check synthetics failure
			messageone-af-s-pla-easier-login-check synthetics failure
			messageone-af-s-pla-email-notifications synthetics failure
			messageone-af-s-pla-login-check synthetics failure
SSL Certificate	alertfind.com	Certificate check	Expiration < 5 days
	chicago.alertfind.com		
	demo.alertfind.com		
	london.alertfind.com		
	plano.alertfind.com		
	virginia.alertfind.com		

ALERTFIND - “AFTER” MONITORING

We started from the AlertFind C4 DS (link) and SaaS DS (link)

ALERTFIND - INFRASTRUCTURE MONITORS			
INFRASTRUCTURE CLASS	**INFRASTRUCTURE ELEMENTS**	**IMPLEMENTATION**	**DEFAULT METRIC AND THRESHOLD**
Compute	central-eks	CloudWatch Agent DaemonSet	cluster_failed_node_count > 0 for 5 minutes pod_cpu_utilization > 99% for 5 minutes pod_cpu_utilization_over_pod_limit > 5% for 5 minutes pod_memory_utilization > 90% for 5 minutes pod_memory_utilization_over_pod_limit > 10% for 5 minutes
	docker-linux-eu-1-dh	CloudWatch Agent	Statuscheckfailed >=1 for 5 minutes cpu_utilization > 99% for 5 minutes **(DD)** disk_used_percent > 90% for 5 minutes mem_used_percent > 90% for 5 minutes
	docker-linux-1-dh		
	docker-linux-2-dh		
Database	central-postgres01	CloudWatch	CPUUtilization > 99% for 5 minutes SwapUsage > 1000000 for 5 minutes ReplicaLag > 3s for 5 minutes DiskQueueDepth > 1000 for 5 minutes

	central-postgres02-cluster	CloudWatch	CPUUtilization > 99% for 5 minutes SwapUsage > 1000000 for 5 minutes ReplicaLag > 3s for 5 minutes DiskQueueDepth > 1000 for 5 minutes
	AWS ElastiCache	CloudWatch	CPUUtilization > 90% for 5 minutes SwapUsage > 50MB for 5 minutes
Storage	AWS EFS	CloudWatch	Throughput Utilization > 90% for 5 minutes PercentIOLimit = 100% for 5 minutes
Network	Route53	CloudWatch	HealthCheckPercentageHealthy < 100% for 5 minutes HealthCheckStatus < 1 for 5 minutes
	VPC	CloudWatch + VPC Reachability	Availability = false

ALERTFIND - APPLICATION MONITORS			
SERVICE CLASS	**SERVICE INSTANCE**	**IMPLEMENTATION**	**DEFAULT METRIC AND THRESHOLD**
Application Service	Custom Fields	CloudWatch Synthetic Canaries	Af_custom-fields_list failed >= 1 in 15 minutes Af_custom-fields_val failed >= 1 in 15 minutes
	Customer Resources		Customer-resources-dc failed >= 1 in 15 minutes Customer-resources-lo failed >= 1 in 15 minutes
	SmartContacts		Af_sc_is_cp_updated failed >= 1 in 15 minutes
	Teams		Af_teams_current-team failed >= 1 in 15 minutes
	Time zone		Af_time-zones failed >= 1 in 15 minutes
	Users		Af_users_get failed >= 1 in 15 minutes Af_users-privileges failed >= 1 in 15 minutes Af_users-count failed >= 1 in 15 minutes Af_users-csv failed >= 1 in 15 minutes Af_users-by-id failed >= 1 in 15 minutes
SSL Certificate	alertfind.com	Certificate check	Expiration < 5 days
	chicago.alertfind.com		
	demo.alertfind.com		
	london.alertfind.com		
	plano.alertfind.com		
	virginia.alertfind.com		