Infrastructure at your Service.

WebLogic Monitoring using RESTful Interface

About me

Pascal Brand

Consultant

+41 79 796 43 59

pascal.brand@dbi-services.com

Who we are
dbi services

Experts At Your Service

- Over 45 specialists in IT infrastructure
- Certified, experienced, passionate

Based In Switzerland

- 100% self-financed Swiss company
- Over CHF6 mio. turnover

Leading In Infrastructure Services

- More than 120 customers in CH, D, & F
- Over 40 SLAs dbi FlexService contracted

Certified Management System SQS ISO/IEC 20000-1

Agenda

1.Introduction

2.Monitoring

3.Conclusion

Introduction

- RESTful General Concepts
- WebLogic RESTful General Concepts
- Monitoring

Introduction
RESTful General Concepts

REST – Representational state transfer

- "It is an architectural style used for web development. Systems and sites designed using this style aim for fast performance, reliability and the ability to scale"
  - Uses HTTP(S) Protocol for data communication
  - Every component is a resource
  - A resource is accessed by a common interface using HTTP standard methods

Introduction
WebLogic RESTful General Concepts

Provides the same view as the Managed Beans tree

- Two main types:
  - Configuration
    - Used to view configuration or configure the WebLogic Server
  - Runtime
    - Used to monitor WLS and for some operations to control WLS
- Available bean trees:
  - Edit access - Administration Server
    - Used to modify the configuration
  - Runtime access - On every server
    - Used to view that server's configuration
    - Access that server's monitoring data
  - Domain access - Administration Server
    - Copies of the runtime beans of all of the running servers
    - Provides a single point of access for monitoring

Introduction
WebLogic RESTful General Concepts

REST URL Format to Manage WebLogic Server

http(s)://host:port/management/weblogic/{version}/path (12.2.1.x)
http(s)://host:port/management/wls/{version}/path (12.1.x)

- where:
  - Host
    The host where WebLogic Server is running
  - Port
    The HTTP or HTTPS port
  - {version}
    The version of a REST resource or latest
  - Path
    The relative path that identifies a resource

Introduction
WebLogic RESTful General Concepts

REST interfaces for accessing resources:

- Configuration
- monitoring

Rest Method	Task
GET	Retrieve data for an object in the tree
POST	Add an object to the tree
POST	Update an object in the tree
POST	Issue a search
DELETE	Delete an object from the tree

Introduction
WebLogic RESTful General Concepts

REST Resource

- Links:
  - Parent
  - Self
  - Childs
  - Operations
  - Create Forms
  - Others

- Items:
  - Properties

Introduction
WebLogic RESTful General Concepts

The WebLogic Server REST Root Resources on Administration Server

URI	Description
/management/weblogic/<version>	Base URI For WebLogic 12.2.1.x REST Accesses
<Base_URI>/serverConfig	View the Administration Server currently running configuration
<Base_URI>/serverRuntime	Monitors the Administration Server
/management/lifecycle	Life cycle management (LCM) REST resources
/management/wls	12.1.3 (legacy) WLS REST resources

Introduction
WebLogic RESTful General Concepts

The WebLogic Server REST Root Resources on Administration Server

URI	Description
<Base_URI>/edit	Edits the WLS configuration
<Base_URI>/domainConfig	View the last activated WebLogic Domain configuration
<Base_URI>/domainRuntime	Monitors the WebLogic Domain
<Base_URI>/domainRuntime/ serverRuntimes	Monitors all the running servers in the WebLogic Domain
<Base_URI>/domainRuntime/ serverRuntimes/<serverName>	Monitors a specific running server in the WebLogic Domain

Introduction
WebLogic RESTful General Concepts

The WebLogic Server REST Root Resources on Managed Server

URI	Description
<Base_URI>/serverConfig	Views the WebLogic Server configuration that a Managed Server is currently running against
<Base_URI>/serverRuntime	Monitors that Managed Server

Introduction
Monitoring

Monitoring is a proactive approach to supervise a system

- Need to make sure that
  - All components of your applications are up and running
  - Your infrastructure and applications do not produce errors
  - Your infrastructure and applications performance matches the SLA

- What to monitor:
  - A lot of metrics are available in WebLogic Servers
  - Select the right metrics and the right thresholds

Introduction
Monitoring ?

Why using RESTful for monitoring ?

- Is lightweight and fast retrieving data for an object in the beans tree
- All you send is a URL request to get the desired WebLogic Server metrics
- The interface is available on administration server and managed servers
- Full support for all WebLogic resources
- Can be used from any programming language
  - UNIX curl, Python, Java, etc..
- No need to run WLST script or JMX on the client side
  - No additional Software installation
- No need to run a WLST locally on the WebLogic Server host
  - No additional resources used

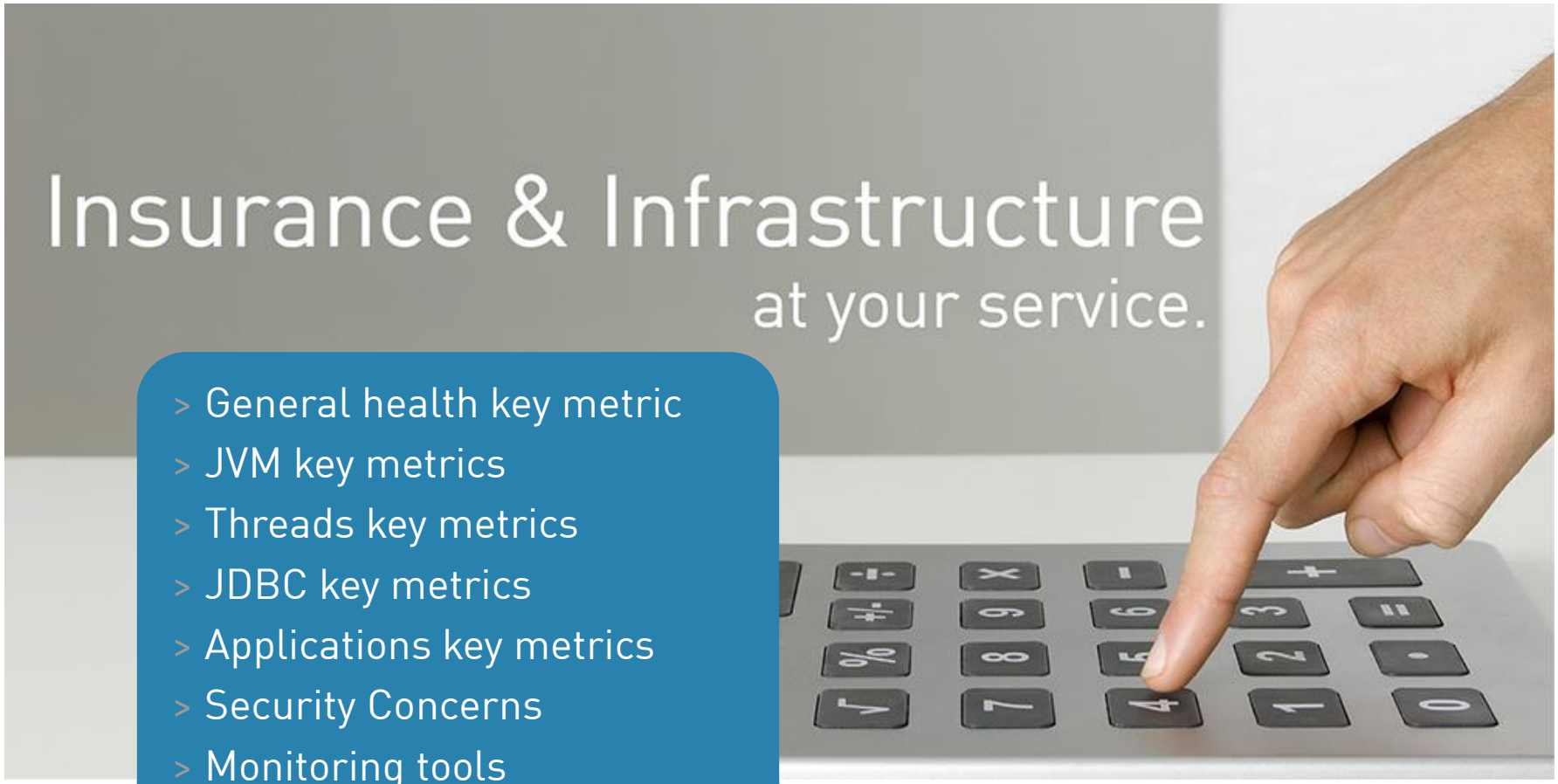
Introduction
Why using RESTful for monitoring ?

The WebLogic Server REST interfaces allows to filter the results

- No additional development needed

Parameter Name	Description
fields	Only return these properties
excludeFields	Return all properties except for these properties
links	Only return links with these names
excludeLinks	Return all links except for the ones with these names

Monitoring



- General health key metric
- JVM key metrics
- Threads key metrics
- JDBC key metrics
- Applications key metrics
- Security Concerns
- Monitoring tools

Monitoring
General Health Metrics

State

- The current Server state

ActivationTime

- Shows the last time the server was started
- Could be crash related, Server has been restarted by Node Manager

Monitoring
General Health Metrics

Get a WebLogic Server General health metrics using WLST

```
userConfig='/app/nagios/etc/objects/scripts/TST_configfile.secure'
userKey='/app/nagios/etc/objects/scripts/TST_keyfile.secure'
address='host01.example.com'
port='7002'
servername='server1'
connect (userConfigFile=userConfig, userKeyFile=userKey, url='t3s://'
+ address + ':' + port)
domainRuntime()
cd('/ServerRuntimes/' + servername )
print ' '
print 'wl_Health_OUT',
    get('State'),
    get('ActivationTime')
disconnect()
exit()
```

Monitoring
General Health Metrics

Get a WebLogic Server General health Key Metrics using 12.2.1 RESTful

```
https://host01:7002/management/weblogic/latest/domainRuntime/serverRuntimes/server1?fields=state,activationTime&links=none
```

```
{
    "state": "RUNNING",
    "activationTime": 1470982524188
}
```

Monitoring
General Health Metrics

Get a WebLogic Domain General health Key Metrics using WLST

```
userConfig='/app/nagios/etc/objects/scripts/TST_configfile.secure'
userKey='/app/nagios/etc/objects/scripts/TST_keyfile.secure'
address='host01.example.com'
port='7002'
connect (userConfigFile=userConfig, userKeyFile=userKey, url='t3s://'
+ address + ':' + port)
domainConfig()
cd('/')
serverNames = cmo.getServers()
domainRuntime()
for serverName in serverNames:
   state(serverName.name,'Server')
disconnect()
exit()
```

Monitoring
General Health Metrics

Get a WebLogic Domain General health Key Metrics using 12.2.1 RESTful

```
https://host01:7002/management/weblogic/latest/domainRuntime/serverRuntimes?fields=name,state,activationTime&links=none
```

```
    {
        "name": "AdminServer",
        "state": "RUNNING",
        "activationTime": 1473234973467
    },
    {
        "name": "server2",
        "state": "RUNNING",
        "activationTime": 1473235510682
    },
    {
        "name": "server1",
        "state": "RUNNING",
        "activationTime": 1473235506258
    }
```

Monitoring
JVM Key Metrics

The memory management is not a task of WebLogic server, this is the work of the Java Virtual Machine

heapSizeCurrent

- The current size (in bytes) of the JVM heap

heapFreeCurrent

- The current amount of memory (in bytes) that is available in the JVM heap

heapSizeMax

- The maximum memory configured for this JVM

heapFreePercent

- Percentage of the maximum memory that is free

Monitoring
JVM Key Metrics

Get JVM Key Metrics using WLST

```
userConfig='/app/nagios/etc/objects/scripts/TST_configfile.secure'
userKey='/app/nagios/etc/objects/scripts/TST_keyfile.secure'
address='host01.example.com'
port='7002'
servername='server1'
connect (userConfigFile=userConfig, userKeyFile=userKey, url='t3s://'
+ address + ':' + port)
domainRuntime ()
cd('/ServerRuntimes/' + servername + '/JVMRuntime/' +
server.getName())
print 'heapfree_' + servername + '_OUT',
    get('HeapFreeCurrent'),
    get('HeapSizeCurrent'),
    get('heapSizeMax'),
    get('HeapFreePercent')
disconnect ()
exit()
```

Monitoring
JVM Key Metrics

Get JVM Key Metrics using 12.2.1 RESTful

```
https://host01:7002/management/weblogic/latest/domainRuntime/serverRuntimes/server1/JVMRuntime?links=none&fields=heapSizeCurrent,heapFreeCurrent,heapFreePercent,heapSizeMax
```

```
{
    "heapSizeCurrent": 259588096,
    "heapFreeCurrent": 101962840,
    "heapSizeMax": 518979584,
    "heapFreePercent": 72
}
```

Monitoring
Threads Key Metrics

ExecuteThreadTotalCount

- This count does not include standby threads and stuck threads.
- The count indicates threads that are ready to pick up new work when it arrives

HoggingThreadCount

- Hogging threads are candidates for stuck threads.
- Might become stuck after StuckThreadMaxTimeoutseconds (default 600secs)

PendingUserRequestCount

- The number of pending user requests in the priority queue. The priority queue contains requests from internal subsystems and users.

Monitoring
Threads Key Metrics

CompletedRequestCount

- The number of completed requests in the priority queue

Throughput

- The mean number of requests completed per second

HealthState

- The health state of this Threads pool

Monitoring
Threads Key Metrics

Get Threads Key Metrics using WLST

```
userConfig='/app/nagios/etc/objects/scripts/TST_configfile.secure‘
userKey='/app/nagios/etc/objects/scripts/TST_keyfile.secure‘
address=‘host01.example.com‘
port=‘7002‘
servername=‘server1‘
connect (userConfigFile=userConfig, userKeyFile=userKey, url='t3s://'
+ address + ':' + port)
domainRuntime ()
cd('/ServerRuntimes/'+ servername +
       '/ThreadPoolRuntime/ThreadPoolRuntime')
print ' ‘
print 'wl_threadpool_OUT',
    get('ExecuteThreadTotalCount'),get('HoggingThreadCount'),
    get('PendingUserRequestCount'),get('CompletedRequestCount'),
    get('Throughput'),get('HealthState')
disconnect ()
exit()
```

Monitoring
Threads Key Metrics

Get The Threads Key Metrics using 12.2.1 RESTful

```
https://host01:7002/management/weblogic/latest/domainRuntime/serverRuntimes/server1/threadPoolRuntime?links=none&fields=executeThreadIdleCount,hoggingThreadCount,pendingUserRequestCount,completedRequestCount,throughput,healthState
```

```
{
    "executeThreadIdleCount": 0,
    "completedRequestCount": 4652,
    "pendingUserRequestCount": 0,
    "healthState": {
        "state": "ok",
        "subsystemName": null,
        "partitionName": null,
        "symptoms": []
    },
    "throughput": 1.999000499750125,
    "hoggingThreadCount": 0
}
```

Monitoring
JDBC Key Metrics

activeConnectionsCurrentCount

- The number of connections currently in use by applications

activeConnectionsHighCount

- Highest number of active database connections in this instance of the data source since the data source was instantiated

waitingForConnectionHighCount

- Highest number of application requests concurrently waiting for a connection from this instance of the data source

waitSecondsHighCount

- The longest connection reserve wait time in seconds

waitingForConnectionFailureTotal

- The cumulative, running count of requests for a connection from this data source that failed to get a connection

Monitoring
JDBC Key Metrics

Get JDBC Runtime Metrics using WLST

```
userConfig='/app/nagios/etc/objects/scripts/TST_configfile.secure'
userKey='/app/nagios/etc/objects/scripts/TST_keyfile.secure'
address='host01.example.com'
port='7002'
servername='server1'
DSname='myDS'
connect (userConfigFile=userConfig, userKeyFile=userKey, url='t3s://'
+ address + ':' + port)
domainRuntime ()
cd ('/ServerRuntimes/' + servername + '/JDBCServiceRuntime/' +
servername + '/JDBCDataSourceRuntimeMBeans/' + DSname)
print ' '
print 'wl_DS_OUT',get('Name'),
      get('ActiveConnectionsCurrentCount'),
      get('ActiveConnectionsHighCount'),
      get('WaitingForConnectionHighCount'),
      get('WaitSecondsHighCount'),
      get('WaitingForConnectionFailure')
disconnect ()
exit()
```

Monitoring
JDBC Key Metrics

Get JDBC Runtime Metrics using 12.2.1 RESTful

```
https://host01:7002/management/weblogic/12.2.1.0/domainRuntime/serverRuntimes/server1/JDBCServiceRuntime/JDBCDataSourceRuntimeMBeans/myDS?links=none&fields=name,activeConnectionsCurrentCount,activeConnectionsHighCount,waitingForConnectionHighCount,waitSecondsHighCount,waitingForConnectionFailure
```

```
{
    "name": "myDS",
    "state": "Running",
    "waitingForConnectionFailureTotal": 0,
    "activeConnectionsHighCount": 3,
    "waitingForConnectionHighCount": 0,
    "waitSecondsHighCount": 0,
    "activeConnectionsCurrentCount": 0
}
```

Monitoring
Application Key Metrics

OpenSessionsCurrentCount

- The current number of opened sessions

SessionsOpenedTotalCount

- The number of sessions handled since the Application start

OpenSessionsHighCount

- Provides the high water mark of the total number of open sessions for this Application

Monitoring
Application Key Metrics

Get Application Runtime Metrics using WLST

```
userConfig='/app/nagios/etc/objects/scripts/TST_configfile.secure‘
userKey='/app/nagios/etc/objects/scripts/TST_keyfile.secure‘
address=‘host01.example.com‘
port=‘7002‘
servername=‘server1‘
appname='myApp‘
connect (userConfigFile=userConfig, userKeyFile=userKey, url='t3s://'
+ address + ':' + port)
domainRuntime ()
cd ('/ServerRuntimes/' + servername + '/ApplicationRuntimes/' +
appname + '/ComponentRuntimes/' + servername + '_/' + appname)
print ' ‘
print 'wl_sessions_OUT',
      get('OpenSessionsCurrentCount'),
      get('SessionsOpenedTotalCount'),
      get('OpenSessionsHighCount')
disconnect ()
exit()
```

Monitoring
Application Key Metrics

Get Application Runtime Metrics using 12.2.1 RESTful

```
https://host01:7002/management/weblogic/latest/domainRuntime/serverRuntimes/server1/applicationRuntimes/SimpleAuctionWebAppDb/componentRuntimes?fields=openSessionsCurrentCount,sessionsOpenedTotalCount,openSessionsHighCount&links=none
```

```
{
    "openSessionsCurrentCount": 12,
    "sessionsOpenedTotalCount": 255,
    "openSessionsHighCount": 15
}
```

Monitoring
Security Concerns

Username & Password provided in the RESTful HTTP Request

- Anyone able to intercept your HTTP traffic will be able to recover the password by simply using an online BASE64 converter

Solutions:
- Use HTTPS for RESTful requests
- Configure the Administration Port
- Run the scripts locally on the Server
- Use a specific user created for monitoring (no editing privileges)

- Curl command line visible in history

Solutions:
- Don't provide the password, curl will request it without echoing it
- Use a encrypted password file

Monitoring

- General health key metric
- JVM key metrics
- Threads key metrics
- JDBC key metrics
- Applications key metrics
- Security Concerns
- Monitoring tools

Monitoring
Tools

Nagios

- Our customers are not running WebLogic 12.2.1.x as their applications are not certified or supported
- Prototype state: Developed little java code
  - username and password provided in encrypted files
  - Bunch of scripts run locally on the machine where the WebLogic Admin Server runs
  - Use https connection
  - See sample call below

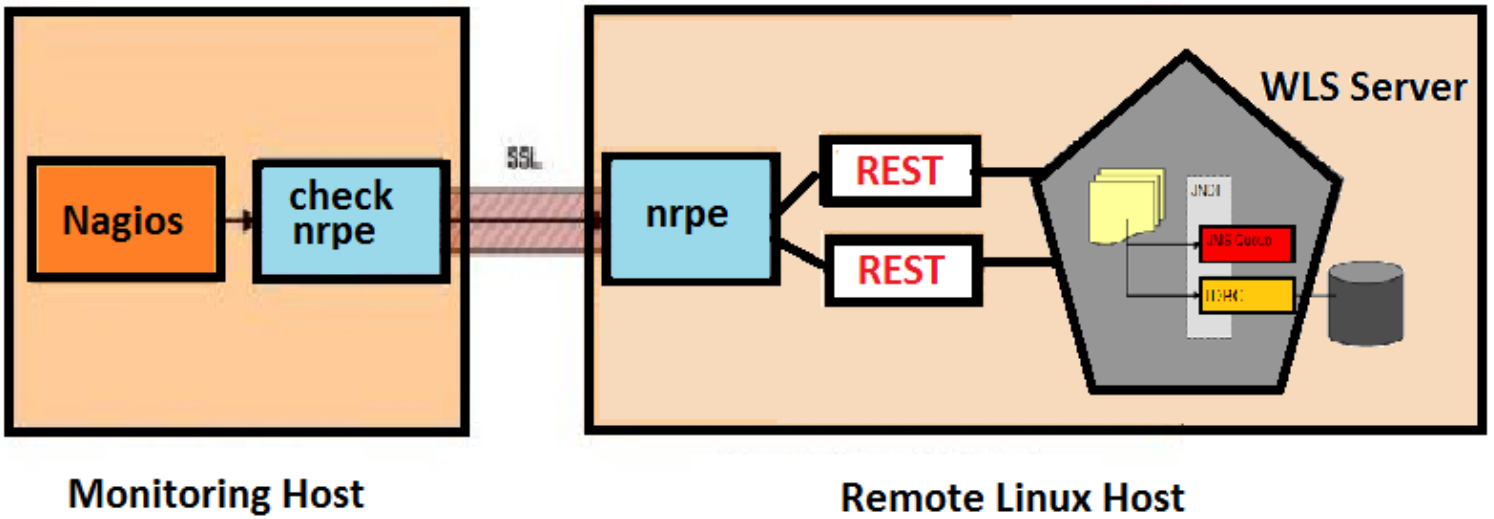
/server/default/deploy/jboss-web.deployer/server.xml

```
[wls@host01 REST]$ java monitoring.rest.WebLogicRest \
https://host01:7002/management/weblogic/latest/domainRuntime/serverRuntimes/server1/JVMRuntime?links=none\&fields=heapSizeCurrent,heapFreeCurrent,heapFreePercent
```

Monitoring
Tools

Nagios

- WLST Replacement with REST calls



Monitoring
Tools

dbi monitoring tool

- Our customers are not running WebLogic 12.2.1.x as their applications are not certified or supported
- Based on Metric definitions created using a GUI interface
  - Services type
  - Target
  - Thresholds
  - Hostnames

Monitoring
Tools

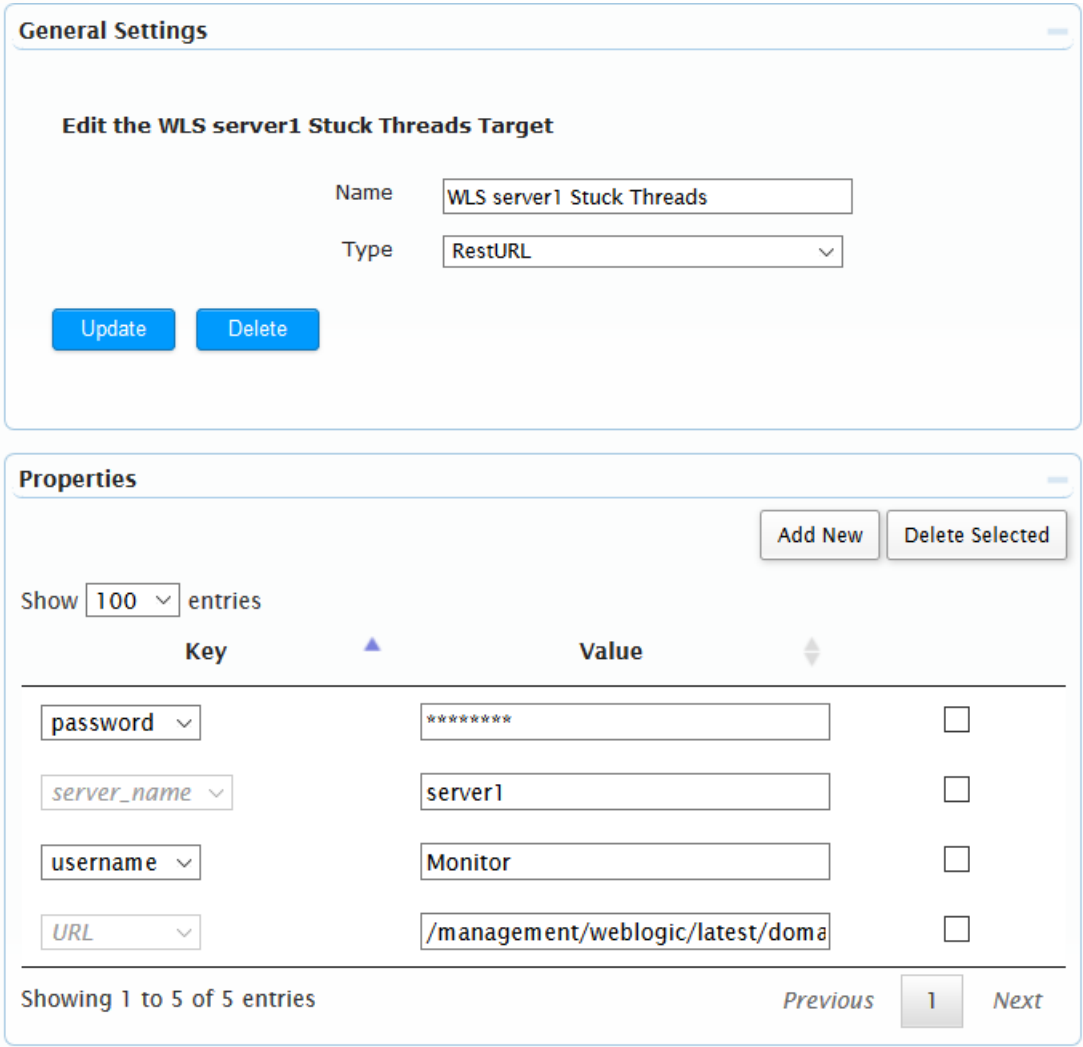
dbi monitoring tool

- Service type base on plugins
  - DFC, Documentum's API
  - URL, HTTP(S) with return code check
  - LOG, report errors analyzing local files
  - WebLogic RESTful
  - And more to come

Monitoring
Tools

dbi monitoring tool

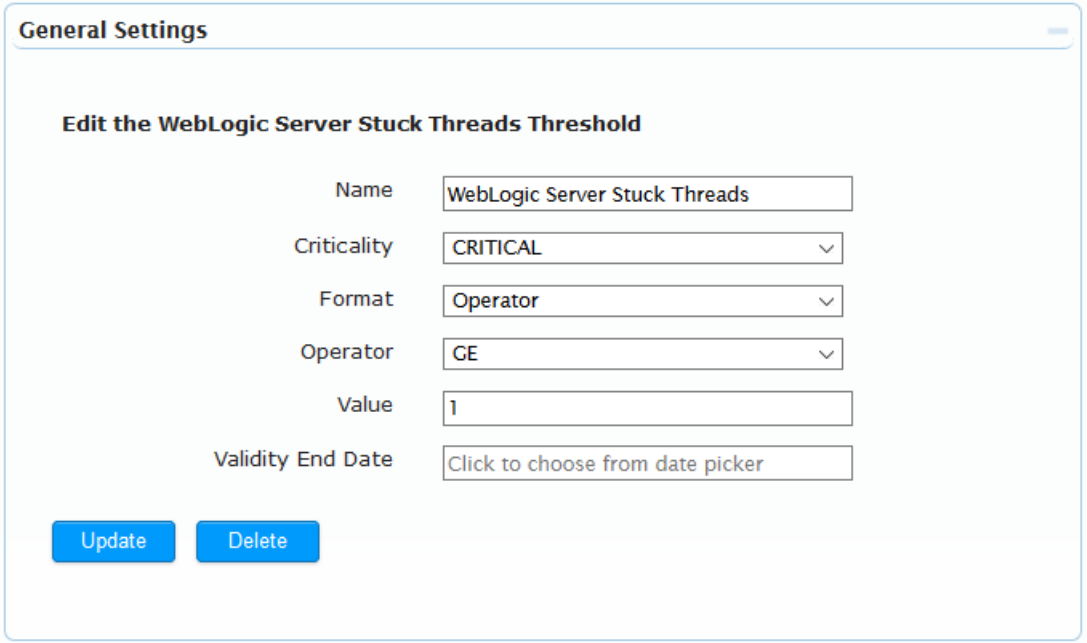
- Target



Monitoring
Tools

dbi monitoring tool

- Threshold



Monitoring
Tools

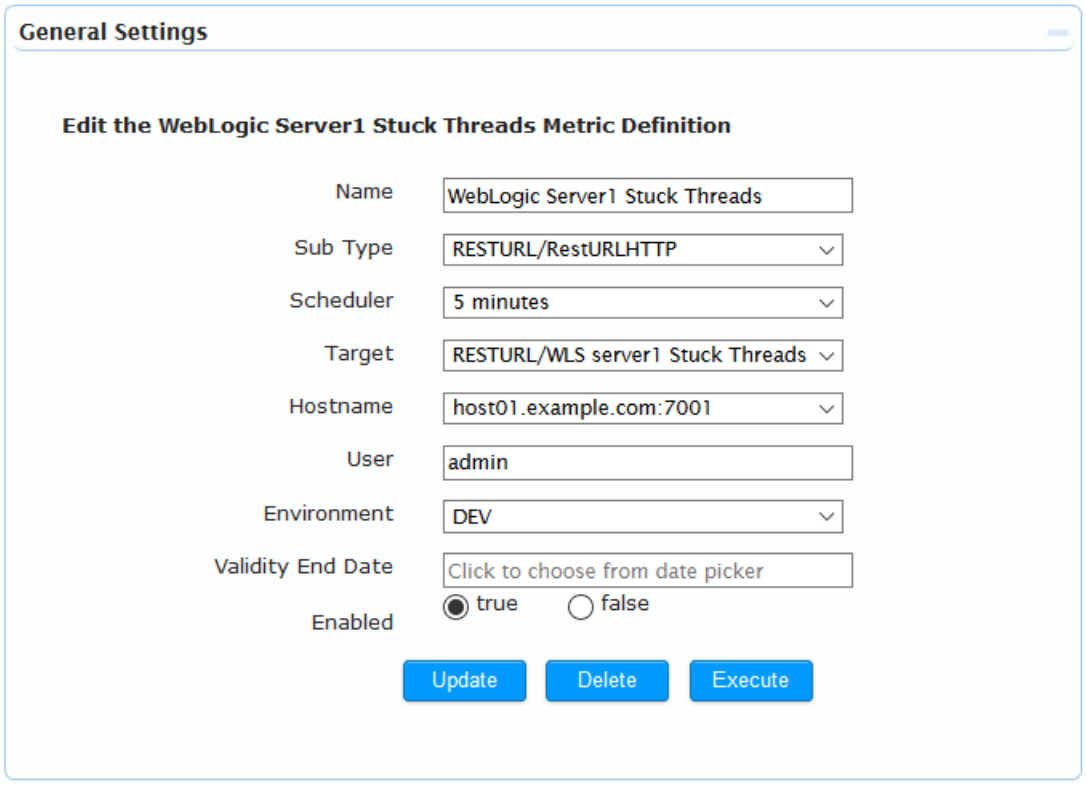
dbi monitoring tool

- Hostname

Monitoring
Tools

dbi monitoring tool

- Metrics



Demonstration

> Demonstration

Conclusion

> Conclusion

Conclusion
Pro & contra

- Lightweight and fast
- No additional software installation
- Easy to integrate with third party software
- No need to start a JVM to run WLST script or run JMX on the client side

- Security Concerns
  - Base64 username password encoding in HTTP Request
- Not fully available before WebLogic 12.2.1.x version

Conclusion
Summary

WebLogic RESTful management is simple and fast

- Easy to interface with monitoring tools

No need for additional development (UNIX curl)

- But if desired can be used from any programming language (Python, Java, etc..)

Any questions? Please do ask

Pascal Brand
Consultant

+41 79 796 43 59
pascal.brand@dbi-services.com

We look forward to working with you!