bischeck

–

A business activity check command server for Nagios

Version 0.3.2
2011-07-29

Contents

1 Introduction **4**

2 Service configuration **6**

3 Date formatting **7**

4 Services and service items classes **8**
4.1 Service classes 8
4.2 Service item classes 8

5 Service item cache - LastStatusCache **9**

6 Threshold configuration **10**
6.1 Twenty4HourThreshold class 10
6.1.1 Period definition 10
6.1.2 Calculation definition 11
6.1.3 Hours and threshold definition 11

7 Data reporting and performance data **13**

8 bischeck configuration files **15**
8.1 properties.xml 15
8.2 bischeck.xml 16
8.3 urlservice.xml 17
8.4 24thresholds.xml 18

9 Service scheduling **21**

10 Configuration tools **22**

11 bischeck internal surveillance **23**

12 Building bischeck **24**
12.1 Jar customization 24
12.2 Developing with bischeck 24

13 Installation **25**
13.1 Upgrading 25

13.2 Getting started . . . 26
13.3 Integration with pnp4nagios . . . 26
13.4 Logging . . . 26

14 Command line utilities **27**
14.1 Run bischeck . . . 27
14.2 List bischeck configuration . . . 27
14.3 Twenty4HourThreshold listing and testing . . . 28

15 Releases **29**
15.1 Release 0.3.2 - 2011-07-29 . . . 29
15.1.1 New features . . . 29
15.1.2 Bugs fixed and important issues . . . 29
15.2 Release 0.3.1 - 2011-04-08 . . . 29
15.2.1 New features . . . 29
15.2.2 Bugs fixed and important issues . . . 30
15.2.3 Upgrade issues . . . 30
15.3 Release 0.3.0 - 2011-03-03 . . . 31
15.3.1 New features . . . 31
15.3.2 Bugs fixed and important issues . . . 31

16 System requirements **32**

17 bischeck license **33**

18 Tips and trick **34**

19 Credits **35**

1 Introduction

bischeck provides business application service checks integrated with Nagios. So what is the difference between application checks and traditional infrastructure checks? Basically its the same, some entity to measure and for that entity define thresholds levels for warning and critical alarms. The demands we have seen in addition to the basic functionality are the following features provided by bischeck:

- Enable a dynamic configuration of threshold values depending of time of the day and day of month or week.
- Support threshold based on fixed values and thresholds based on the measured value from other monitored entities.
- Support multiple scheduling schema per service. This enable a fine grain control of when a service should be run. The configuration is based on a similar structure as unix cron.
- Configure monitored entity that are based on multiple measured entities, what could be described as "virtual" entities.
- Support date macros in execution statements of measured entities, typical used in a where clause when selecting from databases.
- Support a multitude of ways to connected to a the entity to measure by allowing custom service connection methods.
- Support for custom threshold classes to enable any way to define threshold logic.
- Standard integration with nagios over the nsca protocol for passive checks.

Lets look at 3 examples that would be solved by using bischeck:

Example 1 – Monitor the number of orders received during the day. The order management application receive order 24 hours a day during Monday to Friday. The total aggregated number of orders are different depending on time of the day. The business expect to have a total numbers of orders of 1500 at 13:00, at 14:00 the order count should be 2300, at 15:00 it should be 3400, etc. Between every hour we interpolate that the order rate are according to a linear equation. This means that the threshold at 13:20 is (2300-1500)*20/60+1500 = 1767. The warning alarm level should be between 90% and 70% of the threshold and the critical alarm if the measured value should be below 70% of the threshold.

Example 2 – Monitor the number of created invoices in relation to the number of received orders. The invoice system should invoice at lest 80% of the daily incoming orders in the same day with one hour delay. This means that the measured value of orders with one hour delay must be used as a threshold for the number of created invoices.

Example 3 – Monitor the current number of orders and if the inflow is zero we need an alarm. The order system have a table with all received orders, but the requirement is that we need to monitor how many that has been received during the last 10 minutes. If this value is zero an alarm must be generated since its an indication that the sales system are not generating orders. To achieve this monitoring we use the last and the previous sample of the total number of orders from example 1 and create the difference between the two to get a new virtual entity to measure, with an threshold level of 0.

bischeck runs as a stand-alone daemon and communicates with Nagios over the NSCA protocol. bischeck is written in Java and utilize the excellent jsendnsca package to communicate with the nsca daemon on the nagios server. bischeck can of course run on the same server as Nagios or on a remote server.

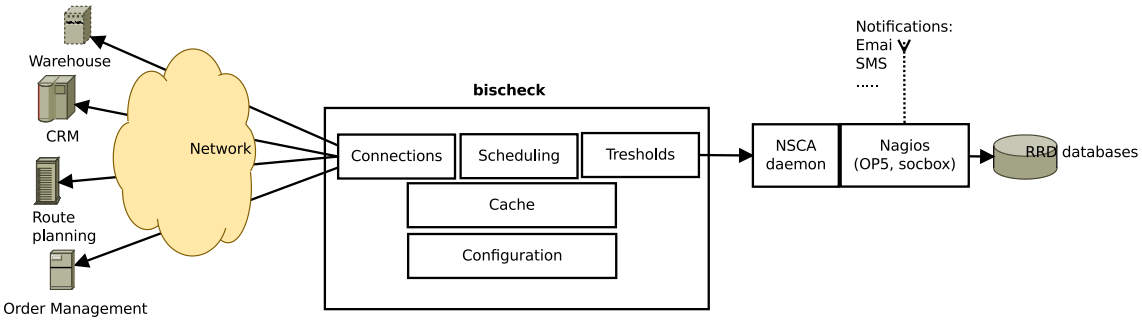

Figure 1.1: Architecture overview

bischeck is open source and licenced under GPL version 2. If you have ideas to new features, find bugs, etc please visit `http://gforge.ingby.com/gf/project/socbox/` where you can fill in bug reports and feature requests. You can also email anders.haal@ingby.com. We look forward to your feedback.

2 Service configuration

Configuration of bischeck is done by describing hosts, services and service items. The host and service names must be the same as you configure in Nagios. Since bischeck work with passive checks the check_dummy or equivalent check command should be used on the Nagios side. The service item describe the entity you want to monitor for the service. As an example we have host erpserver and we define a service called orders. For the order service we can have multiple service items, defined like edi_orders and post_orders. The service define a connection url to connect to the entity to measure and the service items define the statement to be executed to retrieve the entity value. The service url is used by bischeck to instantiate the right service class to manage the service. For example if the url has a schema part that is jdbc, the JDBCService class is used. The mapping between the service url and the class to use is configurable. Each service class must implement the Service interface. The creation of the specific service class is done in the ServiceFactory class.

The execution statement described for the service item must only return one single value. A service item class must implement the interface Serviceitem and is created through the ServiceItemFactory class.
All host, service and service items are described in xml configuration files located in the *etc* directory of the installation.
For more information about services and service items classes please see, chapter 4 on page 8.

3 Date formatting

Many service items that execute statements will typically execute something depending on a date. For this reason bischeck supports date macros in the execute statement string. For example, if this is a sql select statement that using a date condition, the formatting could be done like this:

```
select count(orders) from order where
fromdate='%%yyyy-MM-dd%[M-1]%%' and
todate='%%yy.MM.dd%[D2]%%'
```

bischeck will replace anything between %% and %% with the current date according to the format string. The formatting follows the structure of the java SimpleDateFormatter class. If the format string includes a %[] macro, the current date will be calculated based on the operation. Y means year, M means month and D means day. So D-1 means subtract one day from the current date and M2 means add two months to the current date. A construction like D-1Y-1 is not supported.

4 Services and service items classes

As mentioned earlier the key configuration objects are the service and service item. The service key attributes are the service name that map against the service name on the nagios server, the schedules when to execute the service and the url that define the connection used by the service item.

The service item key attributes are the execute statement that the define what should be processed to retrieve the measured entity and the threshold class used to process the measured entity to validate the state of the measured entity.

4.1 Service classes

The following Service classes is provided in the standard bischeck distribution:

- JDBCService - Manage services connecting to databases over JDBC. JDBC jars should be put in the directory customlib in the bischeck install directory to automatically be found at start-up.
- LastCacheService - Make a connection to bischeck's internal last measured value cache.

4.2 Service item classes

The following ServiceItem classes is provided in the standard bischeck distribution:

- SQLServiceItem - enable execution of a SQL statement. Its important that the SQL statement only return on value, like a "select count(*) ....".
- CalculateOnCache - enable execution using measured values residing in the bischeck internal cache. Mathematical expressions according to jep expressions capability is possible like:

```
if((host1-service1-item1[0] - host1-service1-item1[1]) <
    0,0, host1-service1-item1[0] - host1-service1-item1[1])
```

The above example check is the result of "host1-service1-item1[0] - host1-service1-item1[1]" is less then 0, if yes return 0 else return the result of "host1-service1-item1[0] - host1-service1-item1[1]". The value to use from the cache is defined by an index [X], where 0 is the latest value retrieved for the specific host-service-item.

5 Service item cache - LastStatusCache

bischeck keeps a none persistent cache of the measured service items. For each service item the last 100 measured values are kept in the cache in a LRU list. To retrieve a specific service item from the cache the following format is used, host-service-item[index] where index is the lru entry with 0 specifying that last retrieved measured value , e.g. erpserver-orders-ediorders[0]. The cache content can be used in the specification of execute statements of service items and in specification in threshold classes, see section 6.1 on the next page.

6 Threshold configuration

To define if the service items value retrieved are okay, on a warning level or on a critical level, a threshold class is specified for a specific service item. The threshold object for a service item are stored in a threshold cache and is valid for a period of a day. Every new day the threshold cache is invalidated and bischeck looks for new valid threshold classes to instantiate for the service item for the period of a day.

The reason that the period is set to a day is that there may be a need to configure the threshold object differently depending on the day of the week or month.

A valid threshold class must implement the interface Threshold. The creation of Threshold objects are done through the ThresholdFactory class.

This structure enables a flexible implementation of very different ways to calculate or specify the threshold. In the simplest form a threshold class could just return a constant value and in the more complex solution be based on algorithms, database content, measured values from other service items, day of month or some complex combination.

Warning and critical level specification are also part of the threshold class. The threshold class are also responsible to define what operations are support for the measured value, like measured value should be large, lower or in an interval of the threshold. Warning and critical level are always defined as the percentage of the threshold.

6.1 Twenty4HourThreshold class

The Twenty4Hour threshold class divides the day into 24 hours. For each hour of the day a threshold is defined. The two threshold values that are next to each other used to calculate a slope of a linear equation between the two closest hours. For example, if the threshold value is 1000 at 14:00 and 1600 at 15:00, the calculation for a threshold value between 14:00 and 15:00 is y=x*(1600-1000)/60 + 1000. At 14:20 the threshold is 20*(1600-1000)/60+1000 = 1200.

The threshold model gives a linear equation with one hour granularity, but over 24 hours it can resemble a curve. This behavior is typical in business systems where the key business values are distributed in a none linear and none constant way over the period of a day, e.g. incoming orders over a day.

6.1.1 Period definition

Since the threshold for a service item can be different depending on the month, day of month, week, day of week the configuration supports thresholds to be described on a granularity called periods. A period include multiple months and weeks definitions as long as they share the same threshold definition. For a month it is possible to specify

a specific month and/or a day of a month and for a week a specific week and/or a day of a week. To find the right threshold period the systems look for threshold period specification for a service item in the following order:

1. Month and day of month
2. Week and day of week
3. Day in month
4. Day in the week
5. Month
6. Week
7. Default

Where month is between 1-12, week 1-53, day of month 1-31 and day of week 1-7 (1=Sun). The default threshold period is used if no other matching occurs.

Since holidays are often days where the business are not operational there is a way to describe days that should no have any threshold checks. These exclude days are checked before any other rule above.

6.1.2 Calculation definition

The class support tree ways how the threshold is compered to the measured value:

">" Measured value should be higher then threshold. If the measured value is lower then threshold*warning(%) warning state is set and if measured value is lower then threshold*critical(%) critical state is set.

"<" Measured value should be lower then threshold. If the measured value is higher then threshold*(1-warning(%)) warning state is set and if if measured value is higher then threshold*(1-critical(%)) critical state is set.

"=" Measured should be in the interval of the threshold. If the measured value is lower then threshold*warning(%) OR higher then threshold*(1-warning(%)) a warning state is set and if measured value is lower then threshold*critical(%) OR higher then threshold*(1-critical(%)) a critical state is set.

For complete configuration description see section 8.4 on page 18.

6.1.3 Hours and threshold definition

The measured value is compered against the threshold value that is calculated from linear equation of the two closest threshold values. As described in the introduction of this chapter we can set the threshold values to fixed number. But in a business system this is not enough. Let take an example. The number of orders that can be invoiced

during a day is probably depending on the number of orders received. So instead of setting the threshold to a fixed number we can use an expression based threshold like "80 % of received orders". The syntax of expression based thresholds is simple and powerful. Expressions are based on the JEP package, see chapter 16 on page 32, where the parameters are any measured values that exist in the service item cache, see 5 on page 9. For example the expression *"erpserver-orders-ediorders[0]*0.8"*, the threshold is set to 80% of the last measured value of the service item ediorders for the service order and host erpserver. We could also combine multiple cached values from different sources in the same expression like, *"erpserver-orders-ediorders[0] / geoserver-route-finalroute[0]"* to get some sort of ratio threshold.
If we just need to check parts of a day for thresholds, just set the hour to NULL and no calculation will be done for that time interval. To not do any thresholds checks for a weekend just define rule number 4 for day 1 and 6 with all hours threshold set to NULL.

7 Data reporting and performance data

Since the threshold calculation can vary it is not trivial to define a common format for the messages sent to the remote NSCA daemon. For that reason the Service class is responsible for formatting the output to a format that makes sense to the specific Service class. A default format is provided in the ServiceAbstract class but can be overridden by the Service class implementation. The below description is the default implementation of the ServiceAbstract class.

The data format between bischeck and nagios follow the standard NSCA format. The data is packaged using the methods in jsendnsca. For a service in bischeck the data is split in the plugin output and the performance output. The plugin output has the following format:

<level> <service item name> = <measured value> (<threshold> <warning value> <warning_calc method> <critical value>) <critical calc method> , <service item name> = ...

Since a service can have multiple service items, the output is presented as a concatenated string of the service items. The <measured value> is the value that was retrieved from the execution of the specific service item execute statement. The <threshold> is the current threshold value that the measured value has been compared against. The calculated warning and critical levels are also calculated based on the percentage value of the threshold.

Depending of the calculation method the string representation will differ. The following methods can be support:

- Measure value must be higher the threshold: <warning calc method> = > W > <cirtical calc method> = > C >
- Measure value must be lower the threshold: <warning calc method> = < W < <cirtical calc method> = < C <
- Measure value must be in the interval to the threshold: <warning calc method> = = +-W = <cirtical calc method> = = +-C =

Example of the plugin output:

OK orders = 12000 (11000 > W > 9900 > C > 7700)

In this case the threshold value is 11000 and warning and critical levels has been set to 10% and 30 % of the current threshold. Since the warning and critical level are fixed the calculated level will changed with the current threshold.

If there are no threshold values defined for the current period in which the measured value is done, the threshold is reported as null and no calculation is done that can be

used for notification. Null can also be reported as the measured value. This can occur if a none or a null value is retrieved for the service item, for example from a faulty sql statement.

Service connection problems will be reported as critical. Warning and critical notification will be based on the service item that has the highest level of severity if there are more then one service item defined for a service. If one service item reports critical and the other reports OK the service will report critical.

For the performance data, which nagios graphs (pnp4nagios) are based upon, each service item is included with the addition of the current threshold.

The execution time of the service execute statement is always part of the performance data and reported in milliseconds. With the pnp4nagios template that is provided by the bischeck installation the average execution time is not graphed.

8 bischeck configuration files

The basic configuration file are xml based and located in the *etc* directory of the installation. The distribution package also include all xml schema, xsd, files. For detail information please review the xsd files located in the installation directory *resources*. Remember that all xml configuration files should use HTML encoded characters.

8.1 properties.xml

Holds different bischeck properties. The properties xml has a simpler structure of:

```
<properties>                                    1
  <property>                                    2
    <key>akey</key>                             3
    <value>avalue</value>                       4
  </property>                                   5
</properties>                                   6
```

The following properties should be changed depending on your installation:

- nscaserver - the IP/hostname of the server running the nsca server, e.g. "172.25.1.56", default is "localhost".
- nscaencryption - the encryption used in the transfer, e.g. "XOR", default is "XOR".
- nscapassword - the password for the nsca server, default is "".
- nscaport - the server port for the nsca server, default is "5667".
- cacheclear - the time when the threshold cache should be evicted each day, default is "10 0 00 * * ? *".
- pidfile - the pid file for bischeck, default is "*/var/tmp/bischeck.pid*".

The properties.xml file can also include class specific properties typical used for service item and threshold classes. A class specific property should have a key formatted in the following way - classname.propertyname

Any class specific properties must have a default value implemented by the class itself. E.g. for JDBCService there is a property called querytimeout that sets the max time in seconds before aborting the query. This property has the name JDBCService.querytimeout and have a default value of 10 seconds.

8.2 bischeck.xml

The bischeck configuration is a hierarchy of describing hosts, services and service items to monitor. Each host can have one or more services, and for each service one or more service item can be configured.

```
<bischeck>                                                          1
  <host>                                                            2
    <name>erpserver</name>                                          3
    <desc>ERP server</desc>                                         4
    <service>                                                       5
      <name>orders</name>                                           6
      <desc>Order management</desc>                                 7
      <schedule>0 0/5 * * * ?</schedule>                            8
      <url>                                                         9
        jdbc:mysql://erphost/erpdb?user=bischeck&amp;password       10
           =bischeck
      </url>                                                        11
      <driver>                                                      12
        com.mysql.jdbc.Driver                                       13
      </driver>                                                     14
      <serviceitem>                                                 15
        <name>ediorders</name>                                      16
        <desc>Inbound edi orders</desc>                             17
        <execstatement>                                             18
          select count(*) from orders where createdate=&apos        19
             ;%%yyyy-MM-dd%%&apos;
        </execstatement>                                            20
        <thresholdclass>                                            21
          Twenty4HourThreshold                                      22
        </thresholdclass>                                           23
        <serviceitemclass>                                          24
          SQLServiceItem                                            25
        </serviceitemclass>                                         26
      </serviceitem>                                                27
    </service>                                                      28
  </host>                                                           29
</bischeck>                                                         30
```

In the host section the following elements are defined:

- name - the name of the host. Must be the same name as the host is configured with on the nagios server
- desc - a optional description field

In the service section the following elements are defined:

- name - name of the service. Must be the same name as the configured service on the nagios server
- desc - a optional description field
- schedule - one to many execution schedules can be defined. See the section "Service scheduling" for more information about configuration options.
- url - the connection specification for the server/service to monitor in a url format
- driver - a class name if a specific class is needed by the url specification

In the service item section the following elements are defined:

- name - name of the service item
- desc - a optional description field
- execstatement - the specification of what to be executed to monitor the service item
- serviceitemclass - the ServiceItem class to use for the service item. For ServiceItem classes part of the distribution the class name is enough to specify otherwise the full class name with package should be specified.
- thresholdclass - the Threshold class to use for the service item. For Threshold classes part of the distribution the class name is enough to specify otherwise the full class name with package should be specified.

8.3 urlservice.xml

The configuration fil holds information of the mapping between service url schema and Service class. The urlservice xml has a structure of:

```
<urlservices>                                    1
  <urlproperty>                                  2
    <key>jdbc</key>                              3
    <value>JDBCService</value>                   4
  </urlproperty>                                 5
  <urlproperty>                                  6
    <key>bischeck</key>                          7
    <value>LastCacheService</value>              8
  </urlproperty>                                 9
</urlservices>                                   10
```

- key – the schema part of the service url, i.e. jdbc.

- value – the Service class name. For Service classes part of the distribution the class name is enough to specify otherwise the full class name with package should be specified.

Important is that the name field for host, service and service item are not allowed to include the dash (-) character.

8.4 24thresholds.xml

The threshold class Twenty4HourThreshold described in section (6.1)has a xml based configuration file.

```
<twenty4threshold>                                                  1
  <servicedef>                                                      2
    <hostname>erpserver</hostname>                                  3
    <servicename>shipments</servicename>                            4
    <serviceitemname>outboundshipment</serviceitemname>             5
                                                                    6
    <period>                                                        7
            <!-- valid for any 21th day in the month -->           8
      <months>                                                      9
        <dayofmonth>21</dayofmonth>                                 10
      </months>                                                     11
                                                                    12
      <!-- valid for week 12 (middle of March)                     13
        and if its a Thursday -->                                   14
      <weeks>                                                       15
        <week>12</week>                                             16
        <dayofweek>5</dayofweek>                                    17
      </weeks>                                                      18
                                                                    19
      <calcmethod>&gt;</calcmethod>                                 20
      <warning>10</warning>                                         21
      <critical>30</critical>                                       22
      <hoursIDREF>1</hoursIDREF>                                    23
    </period>                                                       24
                                                                    25
    <period>                                                        26
      <!-- valid if its a Friday -->                               27
      <weeks>                                                       28
        <dayofweek>6</dayofweek>                                    29
      </weeks>                                                      30
      <calcmethod>&gt;</calcmethod>                                 31
      <warning>10</warning>                                         32
```

```
    <critical>30</critical>                                                  33
    <hoursIDREF>2</hoursIDREF>                                              34
  </period>                                                                 35
                                                                            36
  <period>                                                                  37
          <!-- This will be used if no other rule if applicable -->         38
    <calcmethod>&gt;</calcmethod>                                           39
    <warning>10</warning>                                                   40
    <critical>30</critical>                                                 41
    <hoursIDREF>31</hoursIDREF>                                             42
  </period>                                                                 43
                                                                            44
</servicedef>                                                               45
                                                                            46
<hours hoursID="1">                                                         47
  <!-- 00:00 -->                                                          48
  <hour>500</hour>                                                          49
  <!-- 01:00 -->                                                          50
  <hour>1500</hour>                                                         51
  <!-- 02:00 -->                                                          52
  <hour>4000</hour>                                                         53
  ..................                                                        54
  <!-- 21:00 -->                                                          55
  <hour>9000</hour>                                                         56
  <!-- 22:00 -->                                                          57
  <hour>10000</hour>                                                        58
  <!-- 23:00 -->                                                          59
  <hour>11000</hour>                                                        60
</hours>                                                                    61
                                                                            62
<hours hoursID="2">                                                         63
  <!-- 00:00 -->                                                          64
  <hour>1500</hour>                                                         65
  <!-- 01:00 -->                                                          66
  <hour>2500</hour>                                                         67
  <!-- 02:00 -->                                                          68
  <hour>5000</hour>                                                         69
  ..................                                                        70
  <!-- 21:00 -->                                                          71
  <hour>10000</hour>                                                        72
  <!-- 22:00 -->                                                          73
  <hour>12000</hour>                                                        74
  <!-- 23:00 -->                                                          75
  <hour>14000</hour>                                                        76
```

```
  </hours>                                                                  77
                                                                            78
  <hours hoursID="2">                                                       79
    <!-- 00:00 -->                                                         80
    <hour>500</hour>                                                        81
    <!-- 01:00 -->                                                         82
    <hour>erpserver-orders-ediorders[0]*0.8</hour>                          83
    <!-- 02:00 -->                                                         84
    <hour>erpserver-orders-ediorders[0]*0.8</hour>                          85
    ..................                                                      86
    <!-- 21:00 -->                                                         87
    <hour>erpserver-orders-ediorders[0]*0.3</hour>                          88
    <!-- 22:00 -->                                                         89
    <hour>null</hour>                                                       90
    <!-- 23:00 -->                                                         91
    <hour>null</hour>                                                       92
  </hours>                                                                  93
                                                                            94
  <!-- Holidays -->                                                         95
  <holiday year="2011">                                                     96
    <dayofyear>0101</dayofyear>                                             97
    ..................                                                      98
    <dayofyear>1224</dayofyear>                                             99
    <dayofyear>1225</dayofyear>                                             100
  </holiday>                                                                101
                                                                            102
</twenty4threshold>                                                         103
```

The configuration is based on two main parts, a servicedef tag and an hours tag. For each combination of host, service and serviceitem a servicedef tag is specified. Each servicedef can have one to many period specification. The period specify when the threshold is valid, calculation method, warning and critical level, see 6.1.3 on page 11for more information. For a specific period a reference to a specific hours tag must be defined. The hours tag must have 24 hour tags, each represent on hour of the day. An hour tag can have a null value, meaning no threshold, a fixed value or an mathematical expression according to JEP, but with variables from any measured value existing in the last value cache, see 5 on page 9. For more information about threshold specifications please see **??** on page ?? and **??** on page ??. The final tag, holidays, describe any day of the year where no threshold will be tested. For that reason the service will always return a OK state if bischeck services is ran and using the threshold class Twenty4HourThreshold.

9 Service scheduling

The service scheduling enable a service to have multiple scheduling configuration for a single service, but at lest one is mandatory. A scheduling can be described in two ways. The simple format describe a interval execution that are repeated forever. The format is just a number and a indicator defining if the granularity is seconds (S), minutes (M) or hour (H). 10M specify that the service should be executed every ten minutes. The second format is more advanced and follow the cron specification of Quartz, see `http://www.quartz-scheduler.org`. This could be like *"0 15 10 ? * MON-FRI"* which would schedule the service at 10:15am every Monday, Tuesday, Wednesday, Thursday and Friday.

10 Configuration tools

The initial version provides no additional tools for configuration other than the normal editor to manage the xml configuration files. The default configuration files in *etc* directory are according to test example configuration, see 13.2 on page 26.

11 bischeck internal surveillance

bischeck use Java JMX standard for internal monitoring. Please read the javadoc for the following classes to review methods available in:

- ExecuteMBean
- LastStatusCacheMBean

The following JMX settings are used by default and set in the *bischeck* script located in the *bin* directory of the bischeck installation directory.

```
jmxport=-Dcom.sun.management.jmxremote.port=3333
jmxssl=-Dcom.sun.management.jmxremote.ssl=false
jmxauth=-Dcom.sun.management.jmxremote.authenticate=false
```

JMX is only enabled when the *bischeck* script is called with the argument "*Execute -d*", which is the way bischeckd init script call the script *bischeck* to start bischeck in daemon mode.

12 Building bischeck

From version 0.3.2 the build process has changed from buildr to ant. Maybe this is a step in the wrong direction but it was to hard to get buildr to do what we wanted.

Check out the bischeck trunk from gforge.ingby.com:

```
$ svn checkout --username anonymous http://gforge.ingby.com/
   svn/socbox//trunk/src/bischeck
```

To build a bischeck distribution run from the directory where you checked out the bischeck code:

```
$ ant dist
```

This will create a compressed tar file in the target directory, named bischeck-x.y.z.tgz where x.y.z is the version number.

12.1 Jar customization

To support custom jar files please place them in installation directory subdirectory *customlib*. This would typical be jdbc drivers, custom threshold classes, etc.

12.2 Developing with bischeck

Its simple to develop your own service, service item and threshold classes. To develop your own you must follow the interface that exist for each type. For service and service items an abstract class exist with default implementation of most of the methods described in the interfaces.

13 Installation

The latest binary version of bischeck is available on `http://gforge.ingby.com/gf/project/socbox/frs`. To down load click the link or from the command line:

```
$ wget http://gforge.ingby.com/gf/project/socbox/frs/bischeck
   -x.y.z.tgz
```

Un-tar the distribution file in a directory and then run the install script. Make sure you have root privileges doing this.

```
# tar xzvf bischeck-x.y.z.tgz
# cd bischeck-x.y.z
# chmod 755 install
# ./install
# service bischeckd start
```

To get full list of available options to the install script use -u. By default the install script will install bischeck in directory */opt/socbox/addons/bischeck* and with the ownership of the user id *nagios*. Make sure that the user exist before running install.

The last command starts the bischeck daemon with the effective user id of the user id set during install, default nagios.

The process id of the java process running bischeck is located in */var/tmp/bischeck.pid*. This file is used by bischeckd to stop the java program running bischeckd and make sure that only one instance of bischeck is started on the server.

13.1 Upgrading

From version 0.3.2 upgrading is possible. If you already have an installation download the new version as described in the previous chapter but to upgrade run:

```
# ./install -I /opt/socbox/addons/bischeck -X
```

The upgrade will save the current installation in a directory parallel to the new version named *bischeck_x.y.z*, where x.y.z is the version of the old installation.

The file *migrationpath.txt* describe the supported upgrade paths and what migration scripts to run by the install script.

If the upgrade is successful bischeck can be started.

```
# service bischeckd start
```

13.2 Getting started

In the *etc* directory are examples of all the configuration files. These works as an examples to get started with a simple bischeck setup of monitoring a database table. Scripts to create the test database is found in the examples directory and requires Mysql. You do not need a nagios server setup to run the test, just monitor the bischeck log file located by default in */usr/tmp/bischeck.log*. To run the example the mysql jdbc driver is required and the jar file, typical *mysql-connector-java.jar* should be copied to the *customlib* directory to automatically be part of bischecks classpath.

```
$ cd <bischeck install directory>
$ cat examples/create_bischeckverify.sql | mysql -u root
$ cp <some location>/mysql-connector-java.jar customlib/
$ sudo /etc/init.d/bischeckd restart
$ tail -f /usr/tmp/bischeck.log
```

13.3 Integration with pnp4nagios

pnp4nagios can create graph layouts depending on the check command used for the service. Since bischeck is a passive check we need to create a unique check command that match the pnp4nagios layout for bischeck. Create a link in the *libexec* directory on the nagios server:

```
nagios$ ln -s check_dummy check_bischeck
```

When describing the service always use the check_bischeck as the check command. The *check_ bischeck.php* that control the pnp4nagios layout must be copied to the directory *pnp4nagios/share/templates* on the nagios server.

13.4 Logging

bischeck use log4j for logging management. The log4j configuration is described in the log4.properties file located in the resources directory of the bischeck installation. By default bischeck writes log information at level INFO to file */var/tmp/bischeck.log*.

14 Command line utilities

There is a number of command line utilites available in bischeck. All can be ran through the script *bischeck* located in the *bin* directory.

14.1 Run bischeck

The normal way to run bischeck as a daemon using the init.d script bischeckd. It is also possible to start bischeck in continues running mode by executing:

```
$ bischeck Execute -d
```

Running in this way have limitations since the execution will not automatically be placed as background process and the effective user id will be the user starting the process which may not have all permissions according to the installation. Neither will pid files be updated correctly. For production always use the init.d script.

```
$ sudo /etc/init.d/bischeckd start
```

or

```
# service bischeckd start
```

For testing purpose it can be good to just run bischeck once and make sure that every thing is executing as expected. This is done by executing:

```
$ bischeck Execute
```

This will override all scheduling definitions and execute everything directly, but only once.

14.2 List bischeck configuration

To list the current bischeck.xml configuration.

```
$ bischeck ConfigurationManager -l
```

To list all properties in the properties.xml.

```
$ bischeck ConfigurationManager -S
```

To list the url to service mapping in urlproperties.xml.

```
$ bischeck ConfigurationManager -U
```

To only validate if the xml configuration files are valid the following command will return 0 if correct. Use $? to see return status.

```
$ bischeck ConfigurationManager -v
```

To show the pid file for the bischeck daemon running.

```
$ bischeck ConfigurationManager -p
```

14.3 Twenty4HourThreshold listing and testing

To list and validate the 24thresholds.xml configuration file run:

```
$ bischeck threshold.Twenty4HourThreshold -l
```

To verify which threshold rule that will used for a specific host, service and service item for a specific date run:

```
$ bischeck threshold.Twenty4HourThreshold -h erphost -s
   orders -i ediorders -d 20111207
```

The above command will show which threshold definition that will be used for host erphost, service orders and serviceitem ediorders on seventh of December 2012. If -d is omitted the current date will be used for the test.

15 Releases

15.1 Release 0.3.2 - 2011-07-29

15.1.1 New features

- The configuration system has been completely rewritten and now us xml based configuration files. Each configuration file has a corresponding xsd file that can be used for verifications. The dependencies to sqlite3 has been deprecated and is just part of this release to support upgrade.
- The scheduling of services and its related serviceitem(s) has been rewritten to support different scheduling polices per service instead of earlier versions of fixed interval scheduling. With 0.3.2 each service can have one to many schedule tags in bischeck.xml configuration file. For more info please see the chapter 9 on page 21.

15.1.2 Bugs fixed and important issues

- The active attribute on Hosts, Services and Serviceitem has been removed.
- The interface com.ingby.socbox.bischeck.threshold.Threshold has a new signature on the method init(). This method now throws Exception.
- The Service interface has two additional methods, setSchedules() and getSchedules().
- The Service interface has changed the signature of getServicesItems() to return Map instead of HashMap.
- buildr has been replaced by ant as the build management system.

15.2 Release 0.3.1 - 2011-04-08

15.2.1 New features

- The ServiceFactory class now use a property table, urlservice, to map what Service class should be instantiate for a specific url schema. The url schema is the key. The current default mapping are:
  - jdbc -> JDBCService
  - bischeck -> LastCacheService

- The ServiceItemFactory class use an additional field, serviceitemclass, in the items configuration table to determine what ServiceItem class to instantiate.
- Calendar in bischeck follows the ISO 8601 date standard by default. This means that the first day in the week is Monday, day 2 according to java.utilCalendar, and that the first week of the year must have a minimum of 4 days. The importance of this is to get the week numbering correct that is used in the configuration in Twenty4HourThreshold class, but day one (1) in the week is still Sunday when defining the tag dayofweek in 24threshols.xml. The setting can be overidden by setting the properties "*mindaysinfirstweek*" (default 4) and "*firstdayofweek*" (default 2) in the properties.xml file.
- If no threshold class has been specified, null in the thresholdclass field in the items table, bischeck will instantiate the "empty" class DummyThreshold.
- For all class configuration of Service, ServiceItem and Threshold its now possible to specify the class name without the package path if the class is part of the bischeck distribution.
- Clean up of the exception handling process when starting bischeck. Now the execution should not start if there are configuration issues with missing classes for Service, ServiceItem and Threshold.

15.2.2 Bugs fixed and important issues

- N/A

15.2.3 Upgrade issues

- Upgrade by doing a fresh installation, but first save the old installation directory. After saving the old installation do a new install. Then copy the files bischeck.conf and 24threshold.conf from old to new installation dir.
- The field serviceitemclass (varchar(256)) in table items in configuration database bischeck.conf must be manual added and populated with the right Service class name. If corresponding service is jdbc:// set the field serviceitemclass to SQLServiceItem and if the service is bischeck:// set the field to CalculateOnCache.
- To add the column:

```
$ sqlite3 bischeck.conf sqlite> ALTER TABLE items ADD
   COLUMN serviceitemclass varchar(256);
```

- Update the serviceitemclass for all rows in items:
  - sqlite> update items set serviceitemclass='SQLServiceItem' where ....
  - sqlite> update items set serviceitemclass='CalculateOnCache' where ....

- Add the new table url2service in database bischeck.conf.

```
$ cat << EOF | sqlite3 bischeck.conf
drop table IF EXISTS urlservice;
create table urlservice(key varchar(128), value varchar(256));
insert into urlservice values ("jdbc","JDBCService");
insert into urlservice values ("bischeck","LastCacheService");
EOF
```

- Copy all file located in the old installation customlib directory to the customlib directory in the new installation.

15.3 Release 0.3.0 - 2011-03-03

15.3.1 New features

- This is the first binary distribution, but should be regarded as a beta version.

15.3.2 Bugs fixed and important issues

- N/A

16 System requirements

bischeck should run on any operating system that supports java. Tests have been done with Java 6. The installation script and init scripts are only supported on Redhat equivalent Linux distributions. Running on none Linux operating system has not been tested.

The following jar packages are distributed as part of the bischeck distribution. All these packages have their own open source licenses.

- commons-lang-2.5.jar - `http://commons.apache.org/lang/`
  - `http://www.apache.org/licenses/LICENSE-2.0.html`
- commons-cli-1.2.jar - `http://commons.apache.org/cli/`
  - `http://www.apache.org/licenses/LICENSE-2.0.html`
- log4j-1.2.16.jar - `http://logging.apache.org/log4j/Apacheproject`
  - `http://www.apache.org/licenses/LICENSE-2.0.html`
- quartz-2.0.1.jar - `http://www.quartz-scheduler.org/`
  - `http://www.apache.org/licenses/LICENSE-2.0.html`
- jsendnsca-2.0.1.jar - `http://code.google.com/p/jsendnsca/`
  - `http://www.apache.org/licenses/LICENSE-2.0.html`
- slf4j-api-1.6.0.jar - `http://www.slf4j.org/`
  - `http://www.opensource.org/licenses/mit-license.php`
- slf4j-log4j12-1.6.0.jar - `http://www.slf4j.org/`
  - `http://www.opensource.org/licenses/mit-license.php`
- jep-2.3.1.jar - `http://sourceforge.net/projects/jep/`
  - `http://www.gnu.org/licenses/gpl-2.0.html`

All jar files distributed as part of bischeck are located in the lib directory.

17 bischeck license

bischeck is licensed under GNU license version 2. For more info please visit `http://www.gnu.org/licenses/gpl-2.0.html`

18 Tips and trick

N/A

19 Credits

Thanks to all people who has developed all the great software that bischeck depends on, and especially all who made Nagios and the Nagios community a success.

A special thanks goes to Peter Johansson and his colleagues at DHL Freight Sweden for sponsoring the development and providing ideas and important feedback.