Icinga and Oracle Part2 - Building software

This page will describe how to build Icinga software using Oracle backend and other oracle related dependencies like oci8 PHP extension and Perl DBD:Oracle module

Part1: Icinga and Oracle Part1 - Installing Oracle Software

Part2: Icinga and Oracle Part2 - Building software

Part3: Icinga and Oracle Part3 - Configuration

Part4: Icinga and Oracle Part4 - Monitoring Oracle

Part2 Building Icinga and dependencies

Prerequisites

C compiler suite

- gcc or a similar C compiler suite and utilities (make, ...) installed. A full gnu build system with gdb, libtool, automake,autoconf etc. is recommended. This will make your life much easier. If gnu build system is not native on you box (e.g. non-Linux systems like Solaris), you have to make sure, Environment variables (PATH, LD_LIBRARY_PATH, LDFLAGS) for gnu tools have to be set in front of current values. Usually the "configure" run tries to use libraries from base system. But then it's possible you are going to mix gnu libraries with

system libraries with the same functionality (mostly happens with libiconv and openssl). This may lead to unpredictable errors like SIGSEGV. The only solution I know is to build these libraries again with gcc und make sure that Makefiles created by configure will have the right path order. Otherwise you have to correct this manually within each Makefile.
If a well working build system is not available, please install it now. ***it makes no sense to proceed without this.***

Oracle Environment checks

- Your database is up and running
- You have set Oracle variables in your environment. It depends of your client type See FullClient Environment or InstantClient Environment
- You are able to connect to your database with Oracle sqlplus utility as sys user. See Oracle Connectivity Tests
- You will need a test user (other than sys) on the database with rights to connect and create objects. Often the "system" account can be used

If not, please correct errors first. ***it makes no sense to proceed without this.***

Build ocilib

A prerequisite for using Oracle with Icinga is currently a current version of OCILib, which is a thin layer on top of Oracle Call Interface(OCI) API. All calls from Icinga to Oracle will be handled by this library. Alternatively such kind of programme may take use of Embedded SQL as provided by Oracle Precompiler ProC, which is now part of Oracle Instant Client as well.

Warning
On Unix systems you should build ocilib for yourself from sources on the target machines to solve all dependencies and prevent funny errors while mismatching build and runtime versions when loading precompiled packages. It's important to have the same Oracle subversion installed for build and runtime. Don't try to use Oracle 10.2 linked software with 10.1 Client. The name of the Oracle client library (libclntsh.10.2.so) is recorded into the binary. Don't try to make symlinks to fake the version. It may work, but you may get very crude errors. BTW: The same restrictions applies to the PHP OCI8 Library.To check the version actually linked you can use ldd.

```
ldd <pathto>/libocilib.so.3.9.2
        libclntsh.so.11.1 => /opt/oracle/instantclient_11_2/libclntsh.so.11.1
(0x00002b81da683000)
```

Exception: For unknown reasons the Oracle 11.2 Full and Instant Client comes with 11.1 named libraries, but they are not equal

- download ocilib http://sourceforge.net/projects/orclib/files/OCILIB%20Sources/ and extract it into an build directory. Starting with Icinga1.5 (Aug 2011) you should choose ocilib 3.9.2+. Do not use Version 3.9.0 and 3.9.1. Ocilib versions below may work, but are not supported

```
tar -xvzf ocilib-3.9.2.tar.gz
```

configure

- run configure command and specify needed options. If you don't have root rights, you can use prefix option to install into your home or somewhere else
  - if you have the full client or server installation, you only need to have ORACLE_HOME configured. All needed files should be found.

```
cd ocilib-3.9.2
./configure --prefix=$HOME/tools
```

  - if you have only the instant client installed, you need additional options. You should set ORACLE_HOME as well to the instant client location or set the similar configure option

```
export ORACLE_HOME=/opt/oracle/instantclient_11_2/
cd ocilib-3.9.2
./configure --prefix=$HOME/tools \
--with-oracle-headers-path=$ORACLE_HOME/sdk/include \
--with-oracle-home=$ORACLE_HOME
```

make

- make it, if there are no errors. Simply enter "make"

test with demo

- optional, but recommended: Make and test ocilib demo
  - change Makefile_demo to test without installing lib, then make it

```
cd demo
vi Makefile_demo
# add to LDFLAGS before existing entries -L../src/.libs -locilib, remove
-locilib at the end
# add to INCS -I../include
make -f Makefile_demo demo
```

  - run test if make succeeded. Syntax ocilib_demo <db> <user> <password>. <user> on <db> needs dba rights. Modify the placeholders to fit your system

```
LD_OLD=$LD_LIBRARY_PATH
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:../src/.libs
./demo <connect> <testuser> <testuserpw>
LD_LIBRARY_PATH=$LD_OLD
# go back
cd ..
```

install

- install ocilib to predefined location: prefix/lib(64) and prefix/include

```
make install
```

Build Icinga binaries

There is nothing special, you can follow Icinga Documentation on http://docs.icinga.org/latest/en/quickstart-idoutils.html
But you should know, libdbi and oracle are alternatives and will not build together. Below my way.

- first extract Icinga source code into your build directory

```
tar -xvzf icinga-1.3.0.tar.gz
#change to the newly create subdir
cd icinga-1.3.0
```

- read INSTALL and README files for dependencies
  Icinga needs some libraries. Most of them are standard within Linux. On other systems you need to install these libraries. As root you can

install such libraries with your package manager. Also install the matching **-devel** packages.
If you have built the libraries from source the "configure" needs to know where these libraries can be found.

```
export CFLAGS="-I<pathto_include_h_dir> $CFLAGS";export CPPFLAGS=$CFLAGS
export LDFLAGS="-L<pathto_lib_dir> -l<libname_without_'lib'> $LD_FLAGS"
#sometimes needed for solaris
export LDFLAGS="$LDFLAGS -lnsl -lsec -lsocket"
```

if you want to use embedded perl you should make sure to compile Perl itself with the same build environment because of dependent library versions.See version consideration at the beginning of this page. Usually Icinga will compile but later core dump unexpectedly. I had this experience on my Solaris box.

configure

- simple version, most suitable on a local linux server and ocilib installed into default location as well (install as root)

```
./configure --prefix=/opt/icinga \
--enable-oracle \
--enable-idoutils \
--with-ocilib-lib=/usr/local/lib \
--with-ocilib-inc=/usr/local/include \
--with-oracle-lib=$ORACLE_HOME/lib
```

If ./configure shows the message "We really need an ocilib to link against" you should try to set LD_LIBRARY_PATH to the Oracle lib directory

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
#or
LD_LIBRARY_PATH=$LD_LIBRARY_PATH=<instantclient-directory>
```

- advanced version for non root, non default installation
  if you are not root and/or root is not willing to install your software you can build and install Icinga completely without root rights. Of cource, you have to change the defaults. Because it's very annoing to enter all the options on command line I create an small shell script and start it instead of calling configure directly. There you can also put your special settings for CFLAGS und LDFLAGS as mentioned above without changing your standard environment. I prefer to change at least the prefix. Below is my configure script for Solaris

```
./configure --prefix=/home/adbm/tools10/icinga \
--with-icinga-user=adbm --with-icinga-group=adbm \
--with-command-user=adbm \
--with-command-group=adbm \
--enable-embedded-perl --enable-nanosleep \
--enable-idoutils \
--enable-event-broker \
--with-httpd-conf=/home/adbm/tools10/Apache2/conf \
--with-temp-dir=/home/adbm/prod/monitor/tmp \
--with-htmlurl=/monitor \
--enable-ssl --with-ssl=$TOOLS \
--with-init-dir=$TOOLS/etc/init.d \
--localstatedir=/home/adbm/prod/monitor/var \
--sysconfdir=/home/adbm/prod/monitor/etc \
--enable-oracle
```

- If you have a non default buildsystem also make sure that system library includes(/usr/include, /usr/lib) are at the end of the *FLAGS lines in the Makefile. Configure will often put this at the beginning causing unwanted library versions being taken.

- Unfortunately, you need to edit every Makefile to fulfil the non-root approach, because configure currently only sets defaults for the install commands.

```
#change from -o root -g root (or others) to your private user
INSTALL_OPTS=-o adbm -g adbm
COMMAND_OPTS=-o adbm -g adbm
INIT_OPTS=-o adbm -g adbm
```

- If you are intend to use plugins written using nagios environment variables you may want to get these variables to be prefixed with NAGIOS_ instead of ICINGA_. You can simply change the string in include/macros.h

```
/****************** MACRO DEFINITIONS *****************/

#define MACRO_ENV_VAR_PREFIX                    "NAGIOS_"
```

  Future versions may get a configure switch (#2083)
  Although I had investigated a lot of time to have a clean build environment, on Solaris I ran into more problems which were solved by code changes. Check bugtracker for details and solutions (#1253, #1772).

build

If configure doesn't complain about errors you can start to compile typing

```
make all
```

Sometimes the build will stop with unresolved symbols. This should not happen because this is what configure is for. But configure will check only one well know library function to determine the presence of the library. This is because such error is caused by a version conflict I mentioned before. Try to find out if you have permission to access libraries and header files and there is no architecture mismatch (64bit programme tries to link 32bit library). If none of them applies you (may) have lost the game at this time and should pray that someone else in this world had the same problem before - and a solution publicated.

install

If the build succeeded you can install it. There is no test as known from other tools available.
But you need an icinga OS user. This may be your common user and you changed the INSTALL_OPTS according, if you are not root or a new one. Then you will need root access. Details in http://docs.icinga.org/latest/en/quickstart-idoutils.html Chapter 2. In this example I assume the user exists.

```
make fullinstall
# or separate
make install
make install-idoutils
make install-commandmode
make install-config
make install-init
make install-webconf
```

install oci8 php extension and configure

You will need to install php oci8 extension if you are intend to use icinga-web. I recommend to build it yourself as discussed above. This is a nearly straight-forward step if you have set your environment correctly. You will have at least three choices. Follow the original build instructions.

See links attached.
You will have to make sure that your web server knows about the Oracle environment and may load Oracle libraries. If you are using Apache web server you can force this by setting with adding environment detectives to your httpd sourcefile

```
vi /etc/sysconfig/httpd
# oracle
export TNS_ADMIN=/opt/oracle/admin/network
export LD_LIBRARY_PATH=/opt/oracle/instantclient_11_2
```

option 1:use your system package manager

Your system distribution may offer prebuild libraries. If you are using this way, you have to make sure this will not disturb installations you made before.

option 2: use an existing php stack and build oci8 as additional module

See http://www.php.net/manual/en/oci8.installation.php and http://www.oracle.com/technetwork/articles/technote-php-instant-084410.html. usual installation

```
#first: set oracle environment
#then
yum install php-devel
tar -xvzf oci8-x.x.x.tgz
cd oci8.x.x.x
phpize
./configure
make
make install
```

option 3: build oci8 together with php

oci8 module is included in php sources and can be builded static or dynamic when building php from scratch. I recommand to build it as shared modul to enable easier replacing of this modul if you have a diffenend oracle client version or a newer version of this modul. To build php from sources you need matching apache-devel packages/sources.

```
php-5.3.5> ./configure --prefix=/opt/Apache2 \
--with-config-file-path=/opt/Apache2/conf \
--with-apxs2=/opt/Apache2/bin/apxs \
--with-oci8=shared,$ORACLE_HOME \
--enable-pdo --with-pdo-oci=$ORACLE_HOME
....
```

option 4: use a prebuild php stack with oci8 included like zend server

See http://www.oracle.com/technetwork/topics/php/zend-server-096314.html
It might be necessary to add Oracle environment into zce.rc

```
IC=/opt/oracle/instantclient_11_2
TNS_ADMIN=/opt/oracle/admin/network
export TNS_ADMIN
if [ -z $LD_LIBRARY_PATH ];then
   LD_LIBRARY_PATH=$IC:/lib:/usr/lib:/usr/local/zend/lib
else
    LD_LIBRARY_PATH=$IC:$LD_LIBRARY_PATH:/usr/local/zend/lib
fi
export LD_LIBRARY_PATH
export PATH=$PATH
....
```

change php.ini settings

you need to enable oci8 extension in php.ini(or a file in php.d/ or ext.d/), if you dont builded it statically into php binary. Make sure, "extension_dir" setting is pointing to the directory, where all shared extensions are located. Additional a valid timezone setting is required.

```
date.timezone = Europe/Berlin
;extension_dir=/opt/Apache2/lib/php/extensions/no-debug-non-zts-20090626
extension=oci8.so
```

In some cases you can also add parameters for oci8 module here, but the defaults are fine in general. See OCI8 section in default php.ini file or oci8 module documentation

Webserver needs to be reloaded after changing these settings to activate. Check error.log and php.log for problems.

test it

Create a small php file within your htdocs root to show php configuration

```
vi phpinfo.php
<?php
phpinfo();
?>
```

Call it from your web browser as below or similar for your environment

```
http://<webserver>/phpinfo.php
```

It will show your active PHP configuration.

- look into section "Apache Environment" (if Apache is used). You should find the settings for TNS_ADMIN, PATH and LD_LIBRARY_PATH you made above
- search for a section "oci8". It should be present and show some lines

Now you should create a small script trying to connect to your DB, modify placeholders for your system and call it similarly like above. You should give a special try to connect with ezconnect syntax (//<host>:<port>/<service>). It should return "Connect OK".

```
vi test_oci8.php
<?php
print "OCI-Test:<br>";
$user='<testuser>'; //change it
$pw='<testuserpw>'; //change it
$db='<connect>'; //change it
$dbh=oci_connect($user,$pw,$db);
if ($dbh) {
     print "Connect OK<br>\n";
}else{
     print "Connect Failed<br>\n";
     $err=oci_error();
     print_r ($err);
}
?>
```

Build and install Oracle Perl Module DBD::Oracle

You will need this module if you indent to use perl based Oracle checks (check_oracle_health and others). Most suitable way to get it using direct install with CPAN. Alternatively you may download source packages for DBD::Oracle and dependencies from http://cpan.perl.org.

Set special environment

First you have to set additional variables which are needed during tests of DBD::Oracle. When setting ORACLE_HOME it will help installer to build it correctly also for use with instant client.
ORACLE_DSN and ORACLE_USERID are required. If you don't set ORACLE_USERID_2, the dependent check will be skipped. See documentation for alternative <connect> descriptor syntax

```
[root ~]# export ORACLE_DSN='dbi:Oracle:<connect>' ;#change it
[root ~]# export ORACLE_USERID=<testuser>/<testuserpw> ;#change it
[root ~]# export ORACLE_USERID_2=<testuser2>/<testuser2pw> ;#change it
[root ~]# export ORACLE_HOME=/opt/oracle/instantclient_11_2/ ;#change it
```

use CPAN

I assume your Perl installation is configured for using cpan. If not, please install it first.

```
[root ~]# cpan DBI DBD::Oracle
```

This will install a fresh DBI and DBD:Oracle module and dependent modules if needed. Look at the DBD::Oracle compatibility list, which Oracle Version is supported by which DBD::Oracle. This depends on the version of Oracle Client libraries as well. Current versions supporting only Oracle9R2 and above.

use source

This method follows the standard perl module installation steps

- extract source (currently DBD-Oracle-1.27.tar.gz) into your build directory and change into
- Read README for dependencies and additional configurations steps
- configure source, make it, test it (now will you need the extra environment variables and: you **should** do this!). If it was OK (see known test issues) install

```
perl Makefile.PL [<options>]
make
make test
make install
```

known test issues

There are well known issues related to LOB and Widechar support checks. But you can safely install the module, if you see a test report similar this one:

```
t/23wide_db.t ........... skipped: Database character set is not Unicode
t/23wide_db_8bit.t ...... skipped: Database character set is not Unicode
t/23wide_db_al32utf8.t .. skipped: Database character set is not Unicode
t/24implicit_utf8.t ..... 1/74
#    row 4: DUMP(nch) = Typ=1 Len=2: 0,191
t/24implicit_utf8.t ..... 54/74
#    row 4: DUMP(nch) = Typ=1 Len=2: 0,191
t/24implicit_utf8.t ..... Dubious, test returned 4 (wstat 1024, 0x400)
Failed 4/74 subtests

..

t/31lob.t ............... 1/12 DBD::Oracle::st execute failed: ORA-24813: cannot send
or receive an unsupported LOB (DBD ERROR: OCIStmtExecute) [for Statement "BEGIN ? :=
DBMS_LOB.GETLENGTH( ? ); END;" with ParamValues: :p1=undef,
:p2=OCILobLocatorPtr=SCALAR(0x1eb4bcd8)] at t/31lob.t line 126.
t/31lob.t ............... Dubious, test returned 1 (wstat 256, 0x100)

...

Test Summary Report
-------------------
t/24implicit_utf8.t   (Wstat: 1024 Tests: 74 Failed: 4)
  Failed tests:  33-34, 70-71
  Non-zero exit status: 4
t/31lob.t             (Wstat: 256 Tests: 9 Failed: 1)
  Failed test:  9
  Non-zero exit status: 1
  Parse errors: Bad plan.  You planned 12 tests but ran 9.
Files=30, Tests=2135, 69 wallclock secs ( 1.96 usr  0.22 sys + 28.25 cusr  2.68 csys =
33.11 CPU)
Result: FAIL
Failed 2/30 test programs. 5/2135 subtests failed.
make: *** [test_dynamic] Fehler 255
```

using cpan you have to force install, which will run build and tests again, but now install regardless of errors

```
cpan -f DBD::Oracle
```

test it

- base check for loading module

```
perl -e 'use DBD::Oracle;'
```

- write a small test script to check connect: test_dbi.pl

```
#!/usr/bin/perl
use DBD::Oracle;

$db='<connect>'; #change it!
$user='<testuser>'; #change it
$pw='<testuserpw>'; #change it
$dbh=DBI->connect('dbi:Oracle:'.$db,$user,$pw);
if ($dbh) {
        print "Connect OK!\n";
}else{
        print "Connect Error:".$DBI::errstr."\n";
}
```

- run it

```
perl test_dbi.pl
```

Build nagios plugin suite

For monitoring services and devices you need check programmes. Icinga comes without such programmes. But because it's compatible with Nagios, you can use all their plugins.
Wwhile building plugins you have to make sure it will match your current Icinga installation parameters. In general the steps are as usual: extract, configure, make, install

- download from http://www.nagios-plugins.org/
- extract into your build directory

```
tar -xvzf nagios-plugins-1.4.15.tar.gz
cd nagios-plugins-1.4.15
```

configure

Here is the example with the standard plugin bundle. There is no Oracle specific, but Icinga specific configuration.
Some checks will only build, if their dependencies are present. Example: If you want to use check_snmp you have to install net-snmp packages before configure, otherwise it will not be there.

```
./configure --prefix=/opt/icinga \
--with-cgiurl=/icinga/cgi-bin \
--with-htmlurl=/icinga \
--with-nagios-user=icinga \
--with-nagios-group=icinga
```

On my solaris box I have to follow the advanced example, configure looks like this

```
./configure --prefix=/home/adbm/tools10/nrpe \
--with-cgiurl=/monitor/cgi-bin \
--with-nagios-user=adbm --with-nagios-group=adbm \
--with-trusted-path=/home/adbm/bin:/home/adbm/tools10/bin:/bin:/usr/bin:/usr/openwin/b
in:/usr/sbin \
--without-ipv6 \
--with-fping-command=$TOOLS\bin\fping \
--with-openssl=/home/adbm/tools10 \
--with-ps-command="/usr/bin/ps -eo 's uid pid ppid vsz rss pcpu etime comm args'" \
--with-ps-format="%s %d %d %d %d %d %f %s %s %n" \
--with-ps-cols=10 \
--with-ps-varlist="procstat,&procuid,&procpid,&procppid,\
&procvsz,&procrss,&procpcpu,procetime,procprog,&pos"
```

Additionally I had to remove references to /usr/sfw/lib and -ldce to prevent a core dump.
A special ps command was nessesary because of the standard "pst3" binary will not work without rights to sys group.

make

```
make
```

install

```
make install
```

install additional check plugins

To check remote hosts and Oracle databases we will need additional check plugins. There is a wide range of plugins for nearly each known device, for which the community offers a special plugin. If not, it's really easy to create your own check plugin.

nrpe

NRPE (Nagios Remote Plugin Executor)" is a plugin for Nagios/Icinga and a service on a remote machine executing predefined checks triggered by the monitoring server. See http://docs.icinga.org/latest/en/nrpe.html for details.

```
./configure --prefix=/opt/icinga \
--with-nagios-user=icinga \
--with-nagios-group=icinga \
--with-nrpe-user=icinga \
--with-nrpe-group=icinga \
--with-nrpe-port=7066
```

check_oracle_health

This library of Oracle related checks will help you to track oracle internal health. Homepage http://labs.consol.de/lang/en/nagios/check_oracle_health/

```
./configure --prefix=/opt/icinga \
--with-nagios-user=icinga \
--with-nagios-group=icinga \
--with-statefiles-dir=/opt/icinga/var/spool/check_oracle_health \
--with-mymodules-dir=/opt/icinga/libexec \
--with-mymodules-dyn-dir=/opt/icinga/libexec

make
make install
```

check_logfiles

This check will be used to check Oracle logfiles e.g. alert.log and listener.log

```
./configure --prefix=/opt/icinga \
--with-nagios-user=icinga \
--with-nagios-group=icinga \
--with-seekfiles-dir=/opt/icinga/var/spool/ \
--with-protocols-dir=/opt/icinga/var/spool
```

check_multi

check_multi allows to combine several checks into one call. This will save performance and allows more checks for a single server.
Homepage http://my-plugin.de/wiki/projects/check_multi/discussion

```
./configure --prefix=/opt/icinga \
--with-nagios-name=icinga \
--with-nagios-user=icinga \
--with-nagios-group=icinga \
--with-tmp-dir=/opt/icinga/var/spool/ \
--with-plugin-path=/opt/icinga/libexec

make all
make test
make fullinstall
```

Build icinga-web

Icinga Web is the new web frontend for Icinga. It will make use of database which is filled by ido2db. Installation is decribed in the Icinga documentation chapter 6 (http://docs.icinga.org/latest/en/icinga-web-scratch.html). Because we are talking about Oracle based installation, we will use Oracle for Icinga web repository and ido2db database as well. Configure options --with-db targetting icinga-web repository connection, --with-api* targetting icinga ido2db schema. Oracle needs only the information in db_name (tnsname or ezconfig url), but the framework needs for building a valid dsn the same information for host and port seperately.
In general you should follow the official installation guide. Because there is will no binary created, you can safely configure for mysql and change database afterwards in app/config/database.xml

```
#sample configure for version 1.7
./configure \
--prefix=/opt/icinga-web \
--with-web-user=apache \
--with-web-group=apache \
--with-db-type=oracle \
--with-db-user=icinga_web \
--with-db-pass=icinga \
--with-db-port=1521 \
--with-db-host=localhost \
--with-db-name=//localhost:1521/xe \
--with-api-cmd-file=/opt/icinga/var/rw/icinga.cmd \
--with-api-subtype=icingaOracle \
--with-api-port=1521 \
--with-api-host=localhost \
--with-api-db-prefix='' \
--with-api-db-user=icinga \
--with-api-db-pass=icinga \
--with-api-db-name=//localhost:1521/xe \
--with-icinga-objects-dir=/opt/icinga/etc/objects \
--with-icinga-bin=/opt/icinga/bin/icinga \
--with-icinga-cfg=/opt/icinga/etc/icinga.cfg
```

After that was successfully, you need to install code and http configuration

```
make install
make install-apache-config
```

installation is not complete here, you need to install the icinga-web repository schema before you can use it. see Configuration Icingaweb

continued in Part3 Icinga and Oracle Part3 - Configuration