Erlang Systems

Design and Configuration Process

Document Number: CDOC XXXX

Author: Sean Hinde

Version: 0.1

Date: 8th March 2004

History

Version	Status	Date	Changes	Author
0.1	Draft	8th March 2004	Initial Version	Sean Hinde

References

Table of Contents

1 Erlang Application Structure

Erlang based applications have a structure defined by Ericsson as part of the OTP framework. A complete system is built from an Ericsson supplied runtime system and set of OTP Applications. These OTP Applications will include standard applications supplied by Ericsson (including as a minimum kernel and stdlib) and additional applications written by T-Mobile or 3rd parties. Regardless of the origin of the OTP Applications the same structure must be followed for all systems.

All code for each individual OTP Application is structured in the filesystem and CVS repository following the standard OTP directory structure:

```
app_name/src/
        /priv/
        /doc/
        /ebin/
        /vsn.mk
```

- `src` contains all Erlang source code files (also referred to as modules) plus the `app_name.app.src` and `Emakefile.src` files
- `priv` contains config files, port programs, esp dynamic web page templates etc
- `doc` contains the generated edoc source code documentation
- `ebin` is an empty directory which will contain the compiled Erlang .beam files after building.
- `vsn.mk` is a file containing a single line which contains the OTP application version number

1.1 The `vsn.mk` file

The structure of the vsn.mk file is as below:

```
APP_NAME_VSN = 1.0
```

This version string relates to the overall version of the OTP application. It is automatically included into the appropriate places by the release building routines.

1.2 The `app_name.app.src` file

All applications must have an `app_name.app.src` file. This file should be stored in the src sub-directory of the app_name directory. This is the standard OTP `app_name.app` file which will ultimately be located in the ebin directory and used by the build system to create the overall boot script for the system. The `.src` version is parameterised by the application version number `APP_NAME_VSN` from the `vsn.mk` file. For example:

```
{application, app_name,
        [{description, "Mission Critical Application"},
         {vsn, "%APP_NAME_VSN%"},
         {modules, [app_name,
                    app_name_app,
                    app_name_db,
                    app_name_lib,
                    app_name_server,
                    app_name_sms,
                    app_name_sup,
                    app_name_test,
```

```
                  app_name_web]},
        {registered, [ app_name_sup, app_name_server]},
        {applications, [kernel, stdlib]},
  {mod, {app_name_app, []}}]}.
```

1.3 Adding a new OTP Application

Every OTP application must have an entry created in the Master CVS control file `Modules`. This entry ensures that the correct shell scripts are run after the export part of a system build.

2 Version Management

Version Management is carried out at a number of different levels following the standard Ericsson derived OTP scheme. Individual OTP applications have their own version number and independent upgrade paths. System level versioning is handled via the standard OTP release (`.rel`) file.

The version control system used currently is CVS. CVS has a number of small limitations which drive some of the details of the versioning scheme (particularly tag naming).

2.1 Code Module Level Versioning

All source code files (Erlang modules) must have a vsn attribute included in the form:

```
-vsn('$Id$').
```

This will be expanded by CVS to include all the required version info after ever check in. This vsn Id is used by the build system and also by tools built into the web gui of every application to query currently running code versions.

2.2 OTP Application Level Versioning

The CVS tagging mechanism is used to track released files. Tags are applied at the OTP application level and below.

2.2.1 Format of CVS version tags

Tags are always created in a standard format. When a release of an OTP application level subsystem is made, all files in the `app_name/*` directory must be tagged with a tag of precisely the form `rel-1-0`. The hyphens are important, as is the quantity of numbering levels.

Interim and bug fix releases of the OTP application or files within it must be tagged with a special tag format which hold the base release version and the patch level reached. Files released between up-issues of the overall application version number should therefore be tagged according to the following convention: `rel-1-0-dcr175` where the `-dcr175` part relates to the change request number this change was issued in. The build management tools understand this format and ensure that anyone asking for `rel-1-0` automatically pick up the latest bug fix releases in version 1.0 while warning them that this is what it is doing.

2.2.2 When to increment the OTP Application version number

An OTP Application level subsystem should have its main version number incremented in the following situations:

- When a new system is being released which incorporates this OTP application and there have been any patches released on top of the current version of the application (i.e. any `rel-x-y-dcr-xxx` patches exist for the current `rel-x-y` version.
- When any non-backwards compatible change is made to any external API provided by the OTP application
- When any source code files are added or removed from the application

In other words the only time that the OTP Application version number should not be increased after a code change in one of its modules is when a change or fix is being released for an existing live system and none of the reasons given above apply.

2.3 System Release Level Versioning

The version of an overall system is controlled by a single file called `system_name-1.0.rel`. The `-1.0` part of this name is significant as it defines the overall version of the system. This file also controls the startup order of the OTP Applications within the system.

2.3.1 Format of System Level Release File

The `system_name-1.0.rel` file takes the form of this example:

```
{release, {"SYSTEM_NAME Transaction Server","system_name-1.0"},
 {erts, "5.3"},
 [{kernel,"2.9"},
  {stdlib,"1.12"},
  {sasl, "1.10"},
  {runtime_tools, "1.4"},
  {os_mon, "1.6.1"},
  {snmp, "3.4"},
  {crypto, "1.2"},
  {ssl, "3.0.1"},
  {mnesia, "4.1.4"},
  {inets, "3.0.6"},
  {asn1, "1.4.2"},
  {shlib, "2.0"},
  {connector, "1.0"},
  {xmerl, "0.18"},
  {checksum, "1.0"},
  {audit_log, "1.0"},
  {cron, "1.0"},
  {stats, "1.1"},
  {alarms, "1.0"},
  {metrica, "1.0"},
  {bos, "1.0"},
  {http_mgr, "1.3"},
  {db_backup, "1.0"},
  {tcp_client, "1.1"},
  {xtc_client, "1.0"},
  {grouse, "2.4"},
  {events, "1.1"},
  {app_name, "1.0"}]}.
```

The first line has two parameters - the system name and version, and the Erlang runtime system version.

Each line in the rest of the file lists an OTP application with its version number. Applications will be started in the order they appear in the list. The version number here must match the CVS Tag for the application or the build system will complain. So the tag `rel-1-0` as applied to the CVS repository would match a version number `"1.0"` in this table.

Note that a tag `rel-1-0-dcr175` will also match the version numnber `"1.0"` in this list, and the build system will simply warn you that you are picking up some patches.

3 Documentation

The following documents are required for every system delivered to the customer:

- High Level Design
- Detailed Design
- User Guide
- Release Notes
- Code Documentation

Code documentation should be created in edoc format embedded within the code. Other documents may be created using either MS Word or created as XML files (preferred).

3.1 Code Documentation

Code documentation should be embedded in the source code files and processed with the edoc tool to extract the documentation into a set of web pages for the application.

Each module should have a top level document describing its purpose and place in the OTP application. All exported functions should be documented including @spec type information.

Created documents should be updated after every change. The contents of the doc directory should be tagged excatly the same as the rest of the files associated with the OTP Application (Check - this should not be strictly neccessary, but is useful for online browsing of docs).

3.2 System Documentation - XML based

The objective of the XML based documentation ystem is to reduce the amount of duplication associated with producing the whole set of documentation. The documentation is created as a set of sections which are pulled together in different way to make e.g. a High Level design or Detailed design. The sections to be produced should include:

- Master document including the system meta data
- Introduction to the system
- Database table desctiptions
- Stats Counter Descriptions
- Log File Descriptions
- User Manual for the web based pages associated solely with this application. Each of the other included OTP Applications will include their own contribution to the User Guide of a system.

System documents should be stored in the doc subdirectory of the system_name_rel CVS Module (Check).

Each system shall have a set of XML Source documents produced. These documents will be processed into PDF files and web pages during the system build process (TODO).