



ERLANG/OTP LATEST NEWS


ERLANG USER CONFERENCE 2009

Kenneth Lundin



CONTENTS

- › Release plans
- › New Build Process for Documentation
- › GIT repository
- › New erlang.org WEB-site
- › Native Implemented Functions (NIFs)



RELEASE PLANS


Decided

- › R13B03 to be released on November 23:rd

Preliminary

- › R13B04 in February 2010
- › R13B05 in April 2010
- › R14 in June 2010

© Ericsson AB 2009 2009-10-29



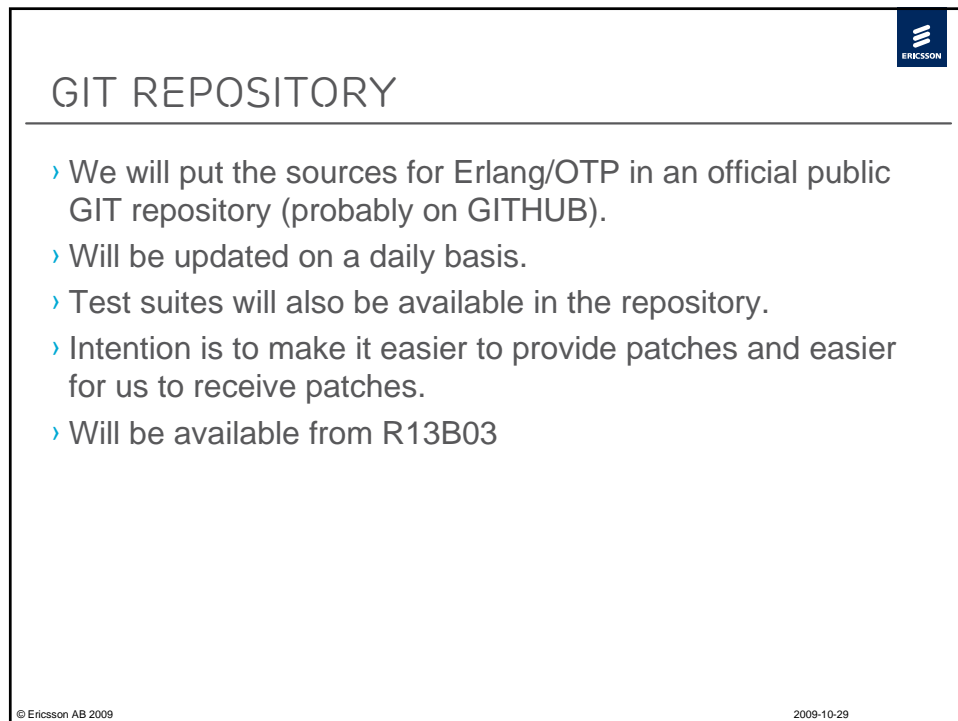
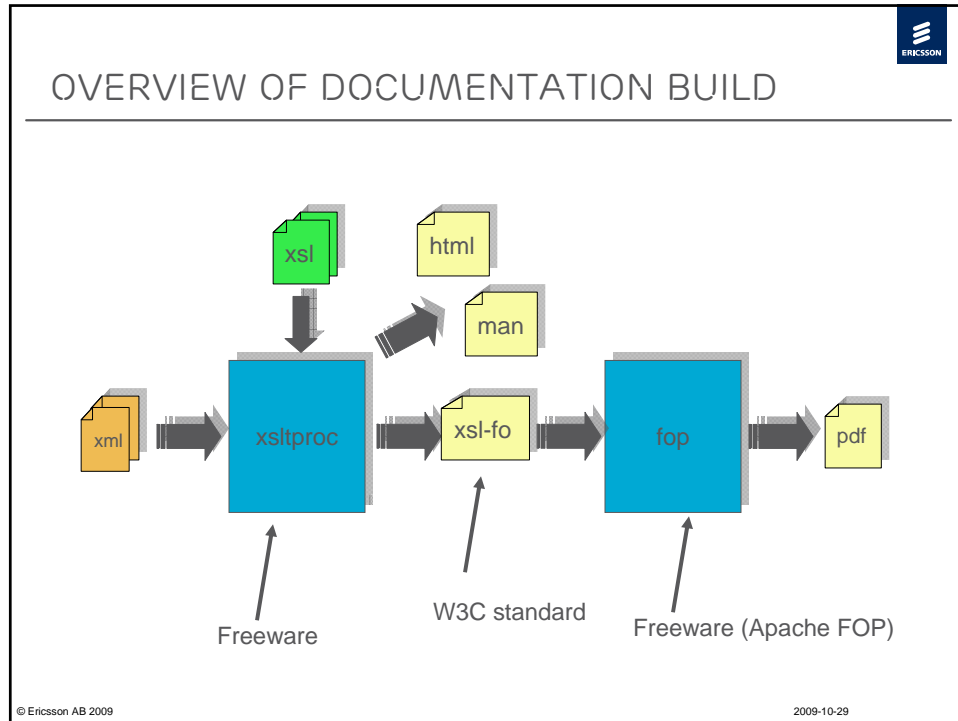
NEW WAY TO BUILD DOCUMENTATION

- › Much faster build
- › Easier to maintain and enhance
- › Produces MAN, HTML and PDF
- › Takes the same XML input as docbuilder
- › docbuilder will be phased out
- › Makes use of well known Open Source tools:
 - `xsltproc` an xslt processor available on all major platforms
 - **Apache FOP** also available on all major platforms

Additional functionality planned in upcoming releases

- › search facilities
- › improved layout.
- › `make doc` should work out of the box (to make it easier for users to contribute to the tools and the documentation)
- › easy to use for everyone documenting their Erlang modules and applications.
- › Integration with edoc
- › new better DTD's and XMLSchemas

© Ericsson AB 2009 2009-10-29






NEW ERLANG.ORG WEB-SITE

- › erlang.org with new layout and technology
- › Easier to update news and articles
- › The goal is to make the site more alive and up to date.




© Ericsson AB 2009
2009-10-29



NEW ERLANG.ORG WEB-SITE (SNAPSHOT)

🔍

[home](#) [news](#) [articles](#) [downloads](#) [documentation](#)




READ ARTICLES >

DOWNLOAD ERLANG/OTP >

DOCUMENTATION >

NEWS & EVENTS

EUC 2009 registration is now open
Written by System administrator, 2009-10-20
 The Erlang User Conference in Stockholm on November 12 is now open for registration. 📅



A new Erlang book is on its way!

GETTING STARTED

Erlang is a programming language used to build massively scalable soft real-time systems with requirements on high availability. Some of its uses are in telecoms, banking, e-commerce, computer telephony and instant messaging. Erlang's runtime system has built-in support for concurrency, distribution and fault tolerance. Originally developed at Ericsson, it was released as open source in 1998.

© Ericsson AB 2009
2009-10-29



NATIVE IMPLEMENTED FUNCTIONS

- › New feature (still experimental) for native implementation of functions (in C)
- › Complementing the driver concept.
- › Exciting, Really Useful, But dangerous
- › We call these functions NIF's (Native Implemented Functions), to differentiate them from BIF's (Built-in Functions) which are more or less part of the language.
- › NIFs offer an easier and more efficient way to implement synchronous functions in C than the driver concept.
- › Dynamically loadable and upgradable
- › Several functions in a module can be implemented in C using this technique. Metadata in the module, the `on_load` attribute, tells the loader which function to call for loading and initialization of the shared library containing the NIF's
- › But as said, really dangerous, **use with care!**

© Ericsson AB 2009

2009-10-29



NATIVE IMPLEMENTED FUNCTIONS (EXAMPLE)

Erlang code

```
-module (niftest).
-on_load(on_load/0).
-export([reverse_bin/1,calls/0]).

on_load() ->
    LibDir = code:priv_dir(myapp),
    erlang:load_nif(filename:join([LibDir,"bin","nifs"])).

%% Dummy implementations
reverse_bin(_) ->
    erlang:error(not_implemented).
calls() ->
    erlang:error(not_implemented).
```

© Ericsson AB 2009

2009-10-29

NATIVE IMPLEMENTED FUNCTIONS (EXAMPLE)



C code (initialization)

```
#include "erl_nif.h"
typedef struct {
    int calls;
} PrivData;

    data->calls++;
}
static int load(ErlNifEnv* env, void** priv_data) {
    PrivData* data = enif_alloc(env, sizeof(PrivData));
    if (data == NULL) return -1;
    data->calls = 0;
    *priv_data = data;
    return 0;
}
static int reload(ErlNifEnv* env, void** priv_data) {
    return 0;
}
static void unload(ErlNifEnv* env, void* priv_data) {
    enif_free(env, priv_data);
}
```

© Ericsson AB 2009

2009-10-29

NATIVE IMPLEMENTED FUNCTIONS (EXAMPLE)




C code (the NIF implementations)

```
static ERL_NIF_TERM reverse_bin(ErlNifEnv* env, ERL_NIF_TERM a1) {
    PrivData* data = (PrivData*) enif_get_data(env);
    ErlNifBinary ibin;
    ErlNifBinary obin;
    int i;

    data->calls++;
    if (!enif_is_binary(a1)) {
        return enif_make_badarg(env);
    }
    enif_inspect_binary(a1, &ibin);
    enif_alloc_binary(ibin.size, &obin);
    for (i=0; i < ibin.size; i++) {
        obin.data[i] = ibin.data[ibin.size-i-1]; /* reverse */
    }
    enif_release_binary(&ibin);
    return enif_make_binary(env, &obin);
}
```

© Ericsson AB 2009

2009-10-29



NATIVE IMPLEMENTED FUNCTIONS (EXAMPLE)

C code (mandatory administration to hook into the Erlang VM)

```
static ErlNifFunc nif_funcs[] =  
{  
    {functionName, arity, fptr},  
    {"reverse_bin", 1, reverse_bin},  
    {"calls", 0, calls}  
};  
ERL_NIF_INIT(niftest, nif_funcs, load, reload, unload)
```

erlangModuleName, funcTable, load_fptr, reload_fptr, unload_fptr

© Ericsson AB 2009 2009-10-29

