Foreword *cont'd*

- QNAP **QPKG**
  - QPKG may be seen as SaaS-like applications
    - Provide a **wide variety of applications** to the end-users
    - Groups of applications are tailored by **market segment** (*usage*)
    - EZ to install/remove by **non-technical end-users**
    - **Generic** (*database, mail*) or **specific** (*ERP*) applications
    - Open system that enables OS **communities to develop QPKGs**
    - Currently all QPKG are **released for free** to NAS users
    - QPKG system will **evolved** in the near future to improve

- QPKG available today
  - Official QPKG
    - MLDonkey, SSOTS (*SlimServer on Turbo Station*), Optware IPKG (*Itsy Package Management System*), phpMyAdmin, Joomla, WordPress, SABnzbd+, AjaXplorer, Tomcat, XDove, Asterisk, JRE 6 (*Java Runtime Environment*),
  - Beta QPKG
    - rTorrent++, NZBGet, IceStation, Plugmedia (*Multimedia Station*), Perl, Piwik, phpBB3, Unrealircd, Q-Ext, Q-Sims (*Virtual World Simulator*), eyeOS, Magento, OpenX, MediaWiki, PostgreSQL

Developer Prerequisite

- In order to develop QPKG applications, you will need to deal with:
  - Linux **Bash** scripts
    - For **installation** and **administration** tasks
  - **PHP** language
    - **Web interface** code writing
    - **JQuery** for Rich Interface Applications
    - **Smarty*** for web page skin management
    - **TinyMCE*** as user text editor
  - Linux **cross-compilation** understanding
    - Compile some application from **source**
  - **CGI** script understanding
    - To issue **admin commands** from the web interface
  - LOCALE and **Internationalization** understanding
    - To enable the QPKG to support several **selectable languages**
    - To enable the QPKG to support **help pages in different languages**
  - **Wiki** page writing
    - To make available online **help** pages
    - To maintain a **troubleshooting** page
    - To provide additional information about the QPKG (eg. case study, technology information, etc…)
  - Technical **English**
    - For **code** comments and QPKG help pages
    - For **Applications Notes** and **Wiki** pages

*

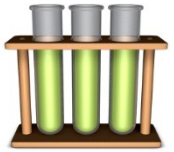
What to package

- There are several levels of packaging:
  - Packaging applications as a PHP **web applications** (*eg Joomla*). The QPKG provides:
    - an installation/remove script to copy/remove the PHP code
    - A folder with the web files
    - an URL link to access the application from the QPKG Interface
  - Packaging applications already existing as **IPKG modules** (*eg Asterisk*). The QPKG provides:
    - an installation script for the IPKG
    - a script to Start/Stop the application
    - An URL link to access the Web server/GUI if any
  - Packaging an already existing **Debian x86/ARMEL package** (*eg Mono*). The QPKG provides:
    - A folder with the binary/libs downloaded from the Deb pkg
    - an installation/remove script to copy/remove the files and recreate the symbolic links to the libs
    - a script to Start/Stop the application
    - An URL link to access the Web server/GUI if any
  - Packaging an application from a **source code** exists (*eg FreeSwitch*). The QPKG provides:
    - A folder with your compiled binaries
    - A folder with the binary/libs downloaded from the Deb pkg
    - an installation/remove script to copy/remove the files and recreate the symbolic links to the libs
    - a script to Start/Stop the application
    - An URL link to access the Web server/GUI if any
  - BUT most of the time you will have to (*eg Plugmedia*):
    - Use some IPKG to **install libs**
    - Compile some **source code** to get the application running
    - Write/modify script for the QPKG **installation/removal**
    - Write/modify script to **start/stop** the application
    - Develop a PHP **interface** to enable the user to configure the application

Internationalization

- Help and Documentation
  - Code files MUST BE documented in **English**
  - User **help pages** shall support Internationalization (*selectable languages*)
  - Application **user interface** (*web GUI*) shall have help pages/pop-ups to provide additional information to the user (*English, Chinese*)
  - **Applications Notes** for each QPKG must be accessible from QNAP website
  - **User documentation**, application examples, troubleshooting sections may be gathered as wiki pages
  - Also consider to translate the application in **Chinese** if not available
    - Translate the *Web GUI*
    - Translate the Help pages any
    - Translate the *Installer* if any
    - Optionally, you can translate (*low priority*)
      - The wiki pages (shall be at least in English)
      - The online help if any
      - The examples
  - Languages (*speakers*)
    - English – mandatory (920 millions)
    - Chinese (1.12 milliard)
    - Hindi (740 millions)
    - French (600 millions)
    - Arabic (482 millions)
    - Spanish (380 millions)

QPKG Debugging



- Full testing requires the QPKG to be created
  - You may install the QPKG manually (see §3)

- Application testing can be done “in situ” by duplication of the QPKG src-xxx folders
  - Copy the relevant src-xxx folders for a specific NAS model under:
    - /share/xx0_DATA/.qpkg/<your_qpkg>
  - Create the symbolic links and set the permissions
  - Run the application from this location

- Installation debugging is tricky
  - Once the QPKG is created copy the QPKG into **/share/Public**
  - Start the installation using **./your_qpkg.qpkg** (*self-extraction*)
  - The QPKG installs on your NAS the same way as if it was uploaded and installed from the QPKG interface
  - Monitor the installation and check for failure

  - **qinstall_xxx.sh** script debugging:
    - There is no easy way to debug the installation script
    - If the qpkg installation fails use **./built.sh <model>** **debug** and check the temporary folder: /mnt/HDA_ROOT/update_pkg/tmp
    - You will find the **qinstall.sh**, **qpkg-name.tgz**, **qpkg.cfg** and **common.sh**
    - Edit and modify the **qinstall.sh** to fix bug
    - Copy the working version back to your SVN

QPKG Debugging

- Recommendations (TBD)
  - Avoid folder removal (rm –R /xxx/$VAR and VAR is empty due to bug or user entry error)
  - Log your code execution
    - into a log file for **debugging** (using a tee instruction for example)
    - Report **critical errors** using the QNAP Admin log system (*/sbin/write_log*)
    - Keep log file **size** under control (tail or rotation files)
  - To improve coding, use "*pair programming*" on sensitive pieces of code
  - Use *code review* across project teams
  - Cross projects with your colleagues for *beta tests*

  - Suggestion : Agile Software development methodology (*e.g. Scrum*)
    - List all tasks into a **backlog**
    - Extract a backlog subset which can provide in a fixed 2 weeks timeslot, a functional* product with the selected features (**Sprint**)
    - **Reiterate** the Sprints until the backlog gets empty

    * *Functional means, all selected features for the Spring have been implemented, tested and validated like in a final product*

QPKG Testing

- Minimum testing to be performed on a QPKG for each NAS models (*series*) before Alpha release
  - Test QPKG installation – qinstall.sh script
  - Test QPKG removal - .uninstall.sh script (*and check for residue after removal such as broken symblink*)
  - Test QPKG Enable/Disable - <your_qpkg>.sh script
  - Test QPKG installation and upgrade
  - Test Web GUI if any
  - Test for security issues if the QPKG needs to support some restrictions (*password, account, multi-users,…*)
  - Test cased can be user cases and may also be providing in the user's document
  - Any other tests to ensure the QPKG to meet the requirements
- If your QPKG (A) is to be used by an other QPKG (B) (eg. PostgreSQL)
  - Test your QPKG (A) with the QPKG (B)
  - Test that the QPKG (A) removal will not destroy data that belong to QPKG (B)
  - For the time being there is no way to prevent the removal of QPKG (A) if QPKG (B) is still installed
- Testing with IPKG modules installation
  - Shall you QPKG use IPKG modules, you must test the correct installation of modules
  - If a pre-installation is enabled (*see next slides for ipkg modules installation*) check for failure and possible remote ipkg installation from Internet
- QPKG shall be released for **Production**
  - Ensure to fully passed the Alpha and Beta levels

PHP Interface Guidelines

- You may have to develop a Application interface
  - This does NOT concern the Application PHP to be packages (*e.g. Joomla PHP or PhpPgAdmin PHP interface*)
  - It is related to YOUR PHP development to create an application configuration page for example.
  - Configuration and variable in the **config.php**
    - No hard coded paths please (*use path variable in config.php*)
    - Variables for : params (Ports, URL, IP Add, ...) etc.
  - Recommended folders structure (*to be discussed first and used by EVERYONE*)

```
├─ /src-shared
│    └─ /<qpkg_name>   (web gui folder)
│         ├─ index.php
│         ├─ /database
│         ├─ /system
│         │    ├─ /_cache                Smarty cache
│         │    ├─ /_compiled             Smarty accelerator
│         │    ├─ /config                configuration files
│         │    ├─ /js                    JQuery, javascript code
│         │    ├─ /libraries
│         │    │    ├─ /smarty
│         │    │    └─ /editors
│         │    │         └─ /tiny_mce
│         │    ├─ /langs                 folder for languages and help files
│         │    ├─ /logs
│         │    ├─ /inc                   folder for php files
│         │    ├─ /themes
│         │    │    └─ /default          Default skin themes
│         │    │         ├─ /css         folder for CSS sheets
│         │    │         ├─ /images      folder for images and thumbnails
│         │    │         ├─ /js          javacript used by the default template only
│         │    │         └─ /tpl         (or inc)
│
├─ / src-xxx
│    └─ /bin          binaries not related to the application (eg. compiled bash) and cgi files
```

Required by Smarty
Required by TinyMCE

PHP Interface Guidelines *cont'd*

- PHP web interface
  - Has DOCTYPE = **DTD XHTML 1.0 Transitional**
    - Needed with JQuery
    - DocType experts are welcomed to address this topic then use the same for all QPKG
  - Has CHARSET = **UTF-8**
  - Is **W3C compliant** – please submit your pages for compliance
  - Has to support **Internationalization** for different languages
    - Set a default language to English using a configuration variable in your config.ini
    - Have a language drop list to enable the user selection
    - Optional : Use browser language detection to find the language or fallback to English
  - Has to follow the **QNAP graphical charter**
    - Get it from QNAP or mimic the v3 admin GUI
    - Share graphics, CSS, Smarty templates between your QPKG devs

- Shall you need a skinner please use **Smarty**
  - Default page size is **1024 x 768** px.
  - Pages must be correctly displayed for this resolution and above.
  - Choose one stable version and keep it for all QPKG

- Shall you need a RIA (*Rich Internet Application*) please use **JQuery**
  - Do not mods the JQuery (*tweek the code*) to allow future upgrades

PHP Interface
User configuration screens

- For application that requires a user configuration
  - Use a QSA – Quick Start Assistant (*Wizard*) and support internationalization
  - 5/6 pages maximum
  - Users are non-technical persons

Q-Sims QSA

PHP Interface
Execute cmds with admin privileges

- PHP web interface
  - User is seen as **Guest** (*Apache*)
  - User cannot issue admin commands using the **exec** PHP instruction

- A workaround exists
  - Call a CGI code (*bash*) from the PHP using the **wget** instruction
  - The CGI code is executed with admin privileges
  - Execution may be incredibly slow because:
    - wget start to check for IPV6 then fallback to IPV4 after timeout
    - Need to force IPV4 (*use a variable to allow easy setup changes*)

- To do
  - Place you cgi files under the right folder:
    - apps/qpkg-name/src-shared – for ASCII cgi files
    - apps/qpkg-name/src-xxx – for compiled cgi files
    - CHMOD 755 qpkg_name.cgi
    - Create a symbolic link from **qpkg_name/bin**/ folder to **/home/httpd/qpkg_name**

- For debugging – unitary tests
  - Enable your CGI to be run from a shell command line

- PHP code
  - Bash command as guest
    $ret = exec('<bash command>', $output);
  - Bash command as admin
    $ret = exec('/usr/bin/wget "http://127.0.0.1:8080/<cgi_file>', $output);
    Please see code examples to log the command output for debugging

PHP interface
Access control

- Basic restricted access control can be implemented
  - By using *.htaccess* file
  - And user from the */mnt/HDA_ROOT/.config/shadow* file
    - Example:

```
AuthName "This Access is Restricted"
AuthType Basic
AuthUserFile /mnt/HDA_ROOT/.config/shadow
AuthGroupFile /dev/null
<limit GET POST>
  require user admin
</Limit>
```

- Protect the user's data
  - Configuration folders must be restricted access
    - Eg .htaccess with deny all
  - Sensitive data must be gathered and stored in a protected folder
  - Protect user's data against typing errors (*confirm deletion/modifications, enable rollback transactions, confirm password change, etc.*)
  - Use poke-yoke to limit user errors or check user entry consistency
    - All default answers are Yes but one which is inverted must be set to No
    - Do not display the 'Continue' button unless all fields are fill-in
  - Implement backup/restore procedures for databases and sensitive data (*.bak files, archive*)

PHP interface
SQL databases

- **MySQL** database system is available on QNAP NAS
  - Recommanded by QNAP
- **PostgreSQL** database system is available through a QPKG
- **SQLite** is not available but through the IPKG modules

Internationalization

- Translated strings must be able to support :
  - HTML tags to improve the presentation
  - Foreign characters, Quotes
- There are mainly two implementations today
  - Use of arrays (*plugmedia*)
    - Loaded as any PHP variable

```
 $lang = array(
'TRIER_PAR'     =>'Trier par',
'ORDRE_TRIS'    =>'Ordre de tris');
```

  - Use of constants (*XDove, q-ext, Q-Sims*)
    - Need to process the language text file (*parser*)
    - Strings are gathered into sections (*e.g. by pages, common, buttons, etc.*)
    -

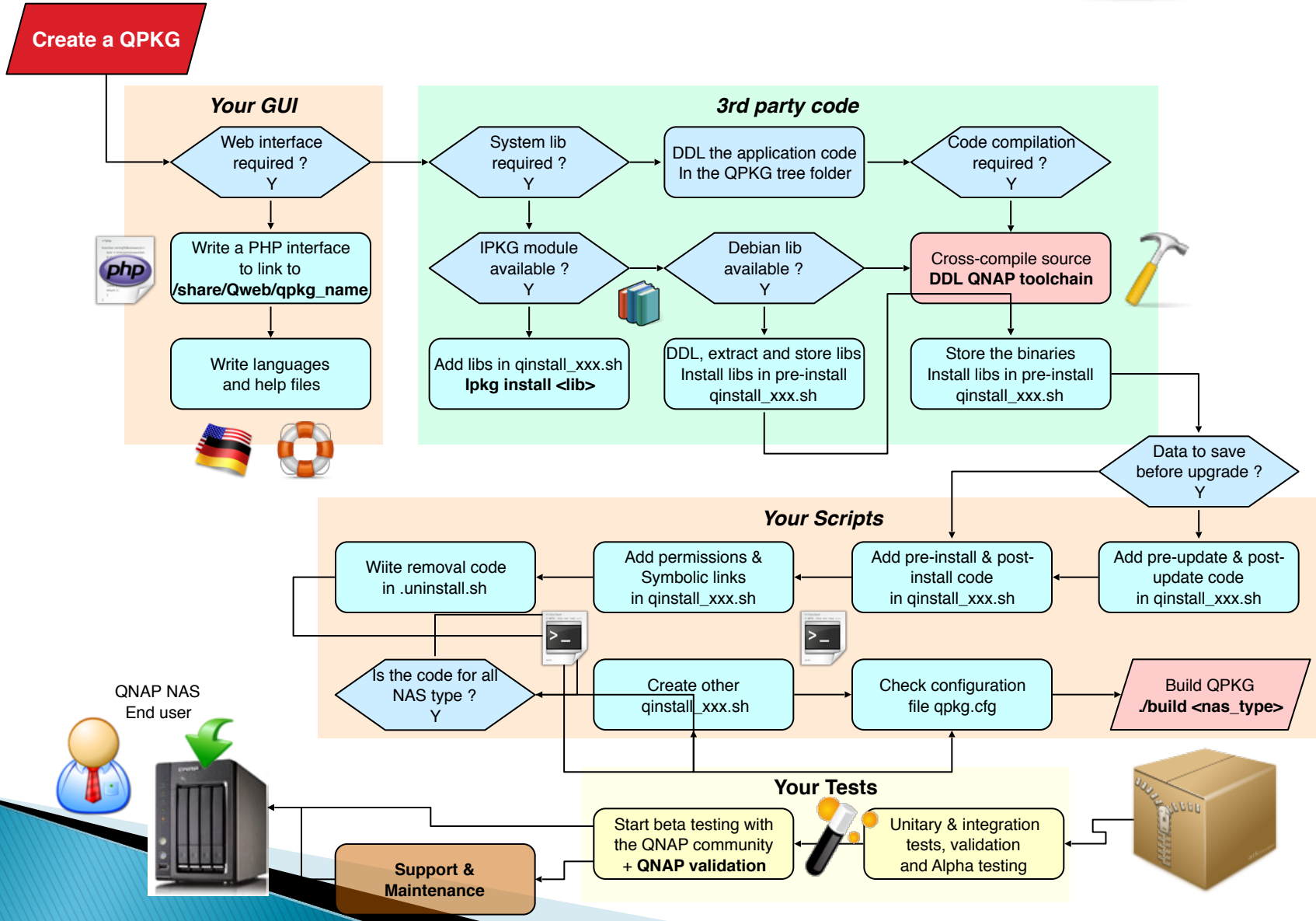
```
[common]
charset                         = iso-8859-15
date_format     = d F Y H:i:s
```

- Translation files for the web interface and help
  - Are 2 different files
  - Translation of the apps interface may be used with default English help pages
  - Do not rely on the LOCALE – NAS are seldom set to correct regionalization
    -

```
# locale -a
C
en_US.utf8
POSIX
```

  - Q-Ext QPKG can install locales if needed but uis still Beta and for Tech. guys

QPKG Dev Sum-up



QPKG Examples

- Some QPKGs (*you must be registered to access the QPKG on the forum*)
  - Forum: http://forum.qnap.com/viewforum.php?f=121
  - Asterisk:
  - XDove:
- QPKGs with QSA (*configuration wizard*)
  - XDove
  - Q-Sims
- QPKG that installs PHP web interface
  - Joomla, phpMyAdmin, AjaXplorer, XDove, PlugMedi@, PostgreSQL, MediaWiki, eyeOS
  - Tomcat,
- QPKG with Ajax
  - PlugMedi@, AjaXplorer
- QPKG that installs binaries
  - JRE, Tomcat, Mono, FreeSwitch
  - SSOTS
- QPKG with admin commands from PHP interface
  - Q-Ext, NZBGet, Q-Sims
- QPKG that have required to compile source code
  - Mono, FreeSwitch
- QPKG that use IPKG pre-installation process
  - PosgreSQL, PlugMedi@ v2

- **WARNING**: QPKGs may have been developed with a different QPKG template and therefore may slightly differ in their folder structure and/or script functions organization