



Implementing InBand Data Transmission with Asterisk

Felix Schupp

Presentation Structure

- About your Presenter
- InBand Data Basics / CTM
- Implementation in Asterisk
- A Real-Life Example
- Work Ahead
- Wrap-Up



About Your Presenter



Felix Schupp

- Founded SoftMethod, Headquartered in Munich since 2004
- Involved with several Open Source Projects, Initiator of the BlackRay Open Source Data Engine (www.blackray.org)
- Vice Chairman of the Open Database Alliance
- Entrepreneur, Technology Evangelist and Sailing Enthusiast.



SoftMethod GmbH

- Focus of SoftMethod is high performance software engineering
- We provide solutions for the telco industry, carriers and contact centers
- SoftMethod has been involved in building Asterisk solutions since 2006
- SoftMethod also offers load testing and technical software quality assurance support.



InBand Data Basics



Why Use InBand Data

- InBand Data refers to the use of a single channel for voice and data
- Modems and Fax are such examples
- InBand Data requires to switch between voice and data modes
- The reliability of voice channels even in mobile networks is very high
- Low bitrate applications can work really well by using the voice channel for data

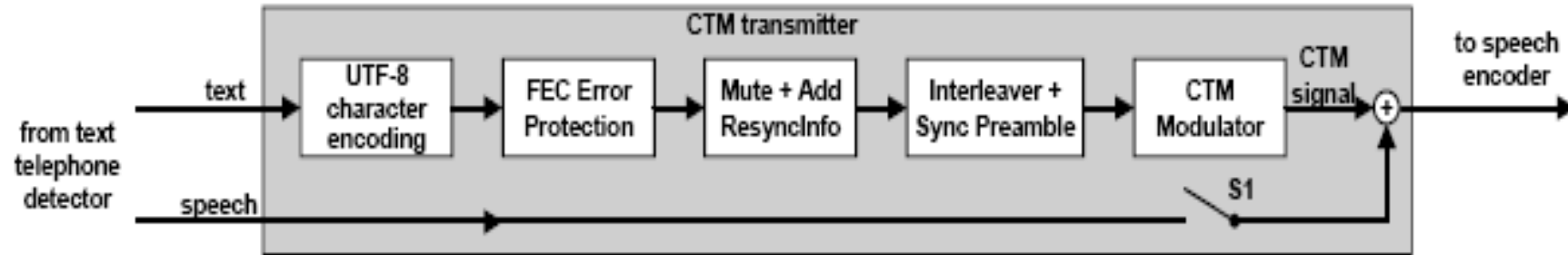


CTM Background

- What is Cellular text Messaging (CTM)?
 - Transmission of text (in-band) via the cellular network
 - Similar to TTY terminals for disabled people but
 - CTM caters for the use in cellular networks
- What is different for cellular networks?
 - Cellular transmission is more error prone (channel fluctuation)
 - Transmission via low-bit rate speech codecs, such as GSM, AMR, etc.



How Does CTM Work?



- Text is converted to an audio signal that is robust for the transmission via cellular networks
- Output signal “looks like” speech so that speech encoders can code it in a meaningful way (do not throw the signal away and replace it by comfort noise)

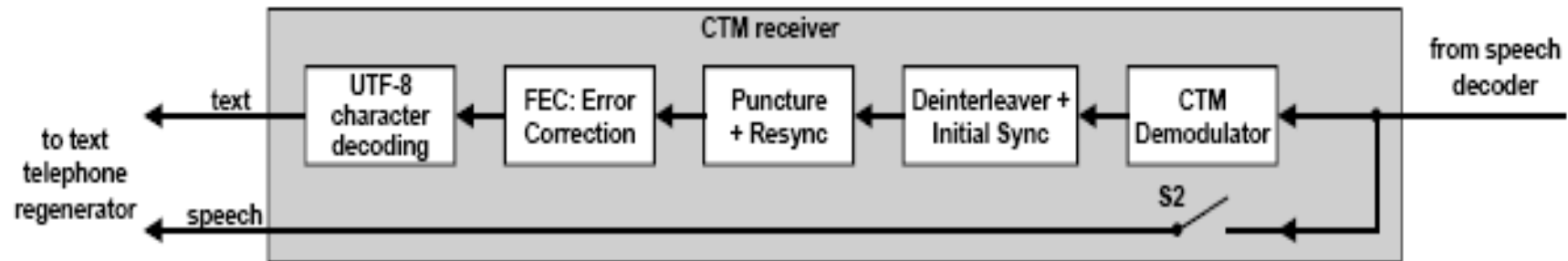


Building Blocks - Modulator

- ASCII to UTF-8 conversion
 - CTM is based on transmission of UTF-8 characters for internationalisation
- Forward Error Correction
 - Convolutional coding ($r=1/4$)
- Mute and AddResyncInfo
 - 80 ms muting intervals every 960ms to prevent non-speech detection
 - training sequence for resynchronisation after an interrupted transmission (e.g. cell hand-over)
- Interleaver + Sync Preamble
 - Provides additional scrambling
 - Preamble to allow detection of the start of a text sequence
- CTM Modulator
 - Uses 4 pilot tones (400Hz, 600Hz, 800Hz, 1000Hz) to transmit 2 bits
 - Tone is sent for 5ms
 - Allows for sending 400 bits/sec



Building Blocks - Demodulator

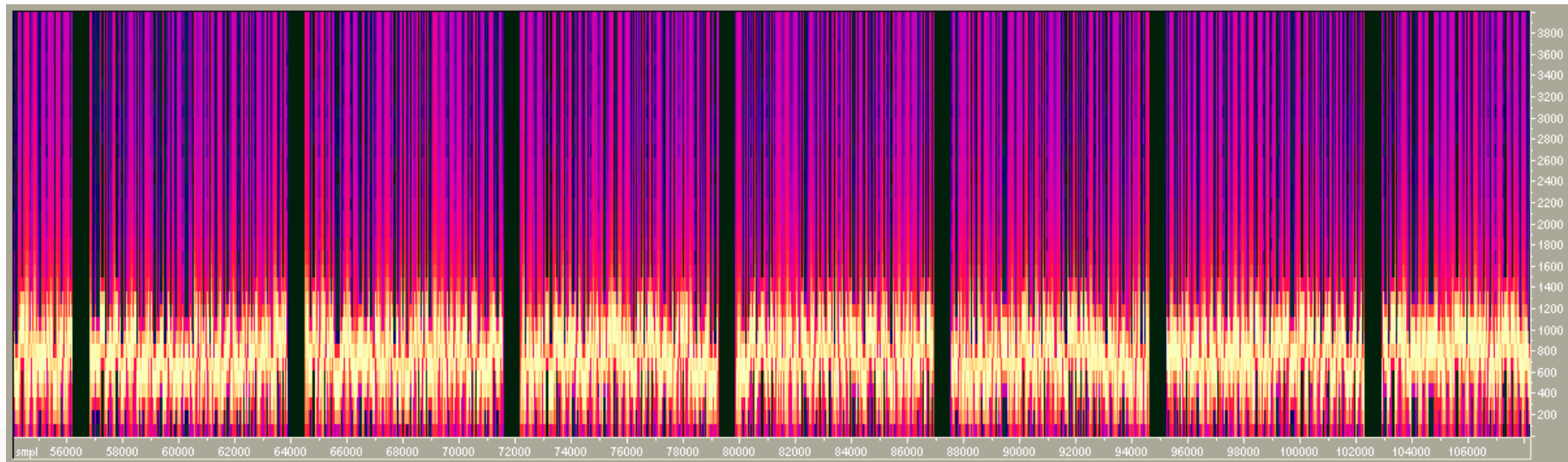


- CTM Demodulator (Possible implementation is given with the spec but can be changed to provide better performance)
 - 4 Band-pass filters feed rectifier and low-pass filters to generate the decoded bits
 - Tracking of signal to have detection in sync (5ms periods)
 - Also to detect whether it contains CTM or speech
- Other building blocks provide the reverse function to the transmitter

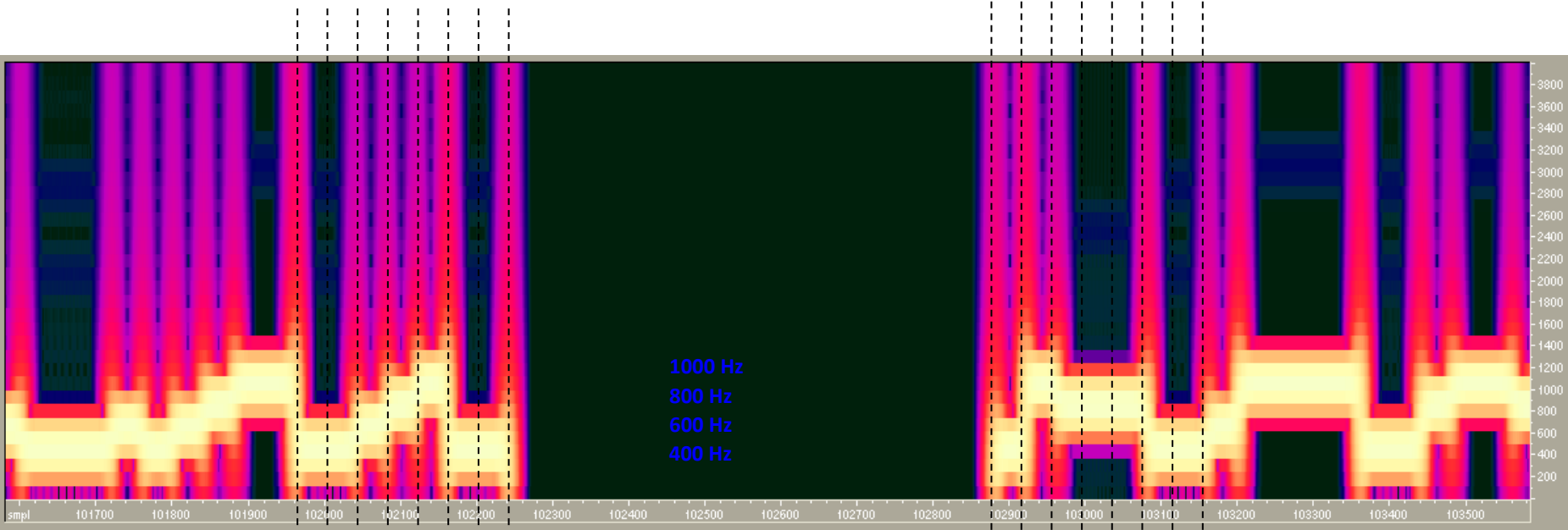


Example Signals

- “Clean” signal after conversion of text to audio signal
 - 16bit linear, 8kHz
- FFT representation of CTM audio signal
 - 32 samples/FFT
- Note: The 80ms muting intervals



Example Signal - Details



Muting Interval

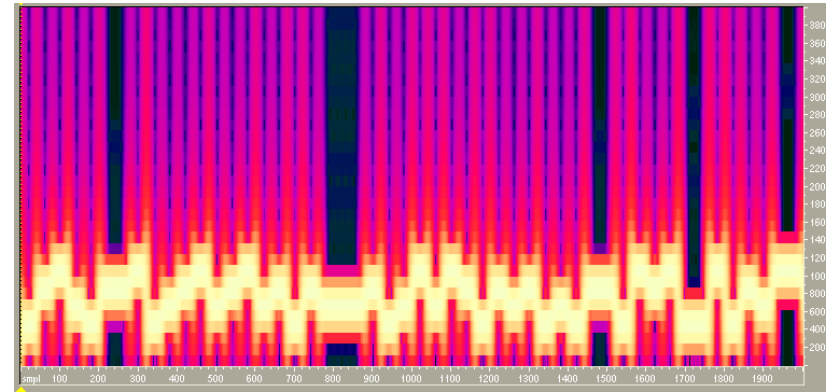
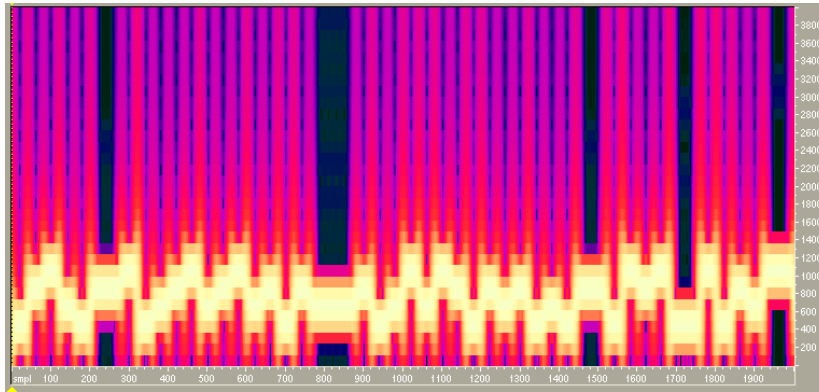
2 bits
2 bits
2 bits
2 bits
2 bits
2 bits
2 bits

2 bits
2 bits
2 bits
2 bits
2 bits
2 bits
2 bits

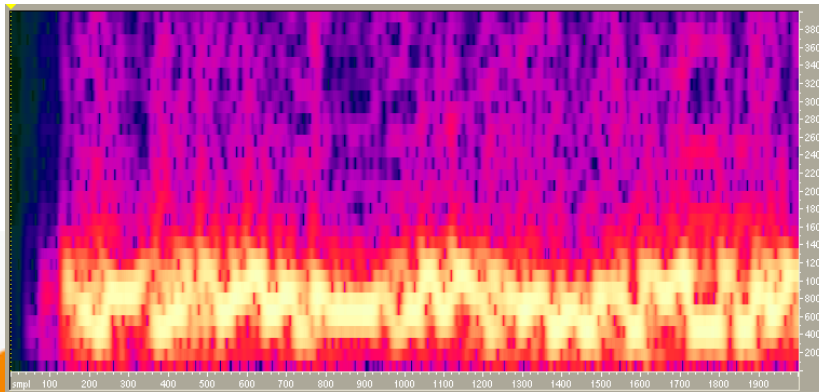


Signal Distortion Due To Speech Codec

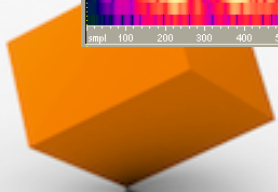
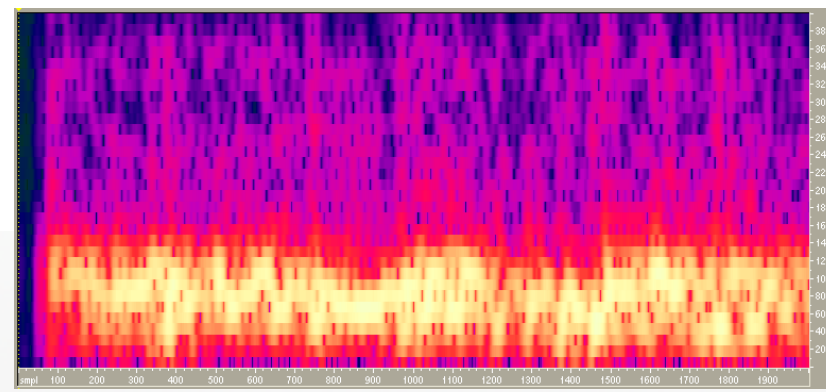
Original Signal



G.729 (8kb/s)



AMR (4.7kb/s)



Channel Capacity

- Modulation bit rate
 - 400 bit/sec
- Actual text transmission bit rate
 - Test file with 4271 bytes transmitted in 411 seconds
 - 10.4 bytes/sec or 83 bits/sec
- Overhead used for
 - Preamble
 - Error Correction (this is the biggest contributor)
 - Muting
 - Resynchronisation



Implementation In Asterisk



Implementation Basics

- The Modem needs to be built as an application, similar to the Monitor application
- Mute/Unmute must be implemented
- The silence-detection and frame elimination optimization need to be modified
- Functions need to be exposed to the Asterisk Manager Interface



Modifications in channel.c

- Similar to how Monitor is implemented, several hooks need to be placed in channel.c
- The modem must always be in line, so it can detect incoming bursts
- Therefore, some of the optimization of not processing frames (muted channel, packets routed directly outside of the channel) must be switched off



The CTM Application Features

- Control of the Modem can be done via the AMI:
 - StartCTM
 - StopCTM
 - SendData
- Events signal InBand Data
 - CTMStarted
 - Mute
 - UnMute
 - DataReceived
 - CTMStopped



Asterisk Manager Interface

- All Methods and Events are available via the AMI
- This allows CTM functionality to be controlled by outside business logic
- CTM can also be used in the dialplan, but it makes more sense via AMI



A Real Life Example

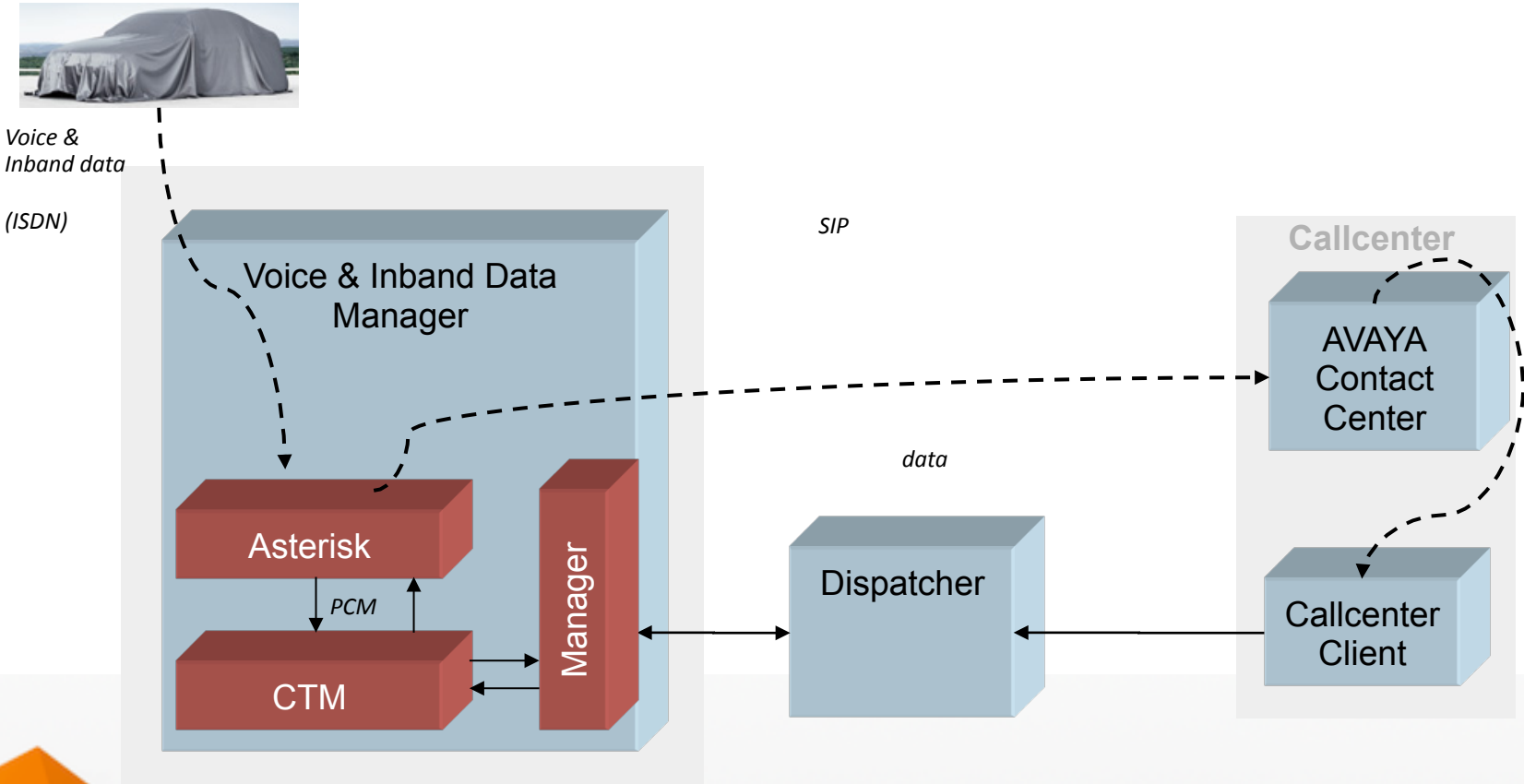


Application Scenario

- The entire solution was built to facilitate the e-Call emergency call requirement on new vehicles
- System uses CTM In-Band data as well as SMS for the service
- Besides the e-Call many other telematic features can be implemented
- Reliability is very crucial in this setup



Use-Case Setup



Work Ahead



Current Development State

- Currently the Implementation is a single Patch against a limited number of Asterisk Versions
- The solution is stable and in pre-production use at a major car maker
- Asterisk Manager Interface support works from Java very well
- The modem is robust but not exchangeable



Modularize The Patch

- The Patch needs to be divided into
 - Basic modifications to channel.c and the manager source
 - A robust interface to make modems dynamically loadable
 - Refactor the modem to become a dynamically loadable module
- This way, other modem implementations can also be plugged in



Implement A Better Modem

- CTM was built for text messaging and is slow
- Better modem implementations will use asymmetrical transmission
- Several studies are available, and could be a basis for new implementations



Add A Flow-Control Layer

- The basic modem implementation works, but a flow control protocol will be required
- This protocol needs to account for the low bandwidth and control things like retransmission and acknowledgements
- A toned-down version of the HDLC framing could be feasible



Add Capability Detection

- Currently, both sides need to be configured the same way
- A better solution would be to perform a better Handshake to exchange basic parameters
- This way different modems can co-exist and only the appropriate one be active



Wrap-Up



What To Do Next?

- Get The Source Code:
 - Register yourself on <http://forge.softmethod.de>
 - SVN checkout available at
 - <http://svn.softmethod.de/opensource/inband/>
- Get Involved
 - Anyone can register and then create tickets, news, participate in the wiki etc
 - We have an active mailing list for discussion
- Contribute
 - We require a signed Contributor agreement before being allowed commit access to the repository



Content License

- This content is licensed under the „Creative Commons Attribution Share Alike 3.0“ License
- <http://creativecommons.org/licenses/by-sa/3.0/>

You are free:

- to Share — to copy, distribute and transmit the work
- to Remix — to adapt the work

Under the following conditions:

- Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

