TCF - Target Communication Framework Update for Helios

Martin Oberhuber
Pawel Piech
Wind River
Motivation: Simple Stacks and Collaboration through an Open Standard
Design Goals of TCF

• **Protocol Framework** provides common infrastructure
  – communication protocol
  – Agent: “Service container”
  – Proxying
• Same protocol on all layers supporting value-add
  – Support pass-through
• Tools can use services in uniform way
• Service implementers can focus on functionality
TCF - Core Design Ideas

• Use the same **extendable protocol** end-to-end
  – allow value-adding servers to intercept select services
• Extension: Abstract Services as building blocks
  – Same tool for multiple targets (e.g. agent, OCD, simulator)
  – Avoid tools specific agents
  – Bridge gap with specific services to configure common ones
• Data-driven by target
  – Service knows best how to represent the system
  – If not possible, put the knowledge in the lowest possible layer and data drive the layers above
  – Auto-discover targets and their capabilities
• Support high latency communication links
TCF: Common agent and protocol
Channels and Messages

• Communication between peers use channels
• Channels abstract/hide the transport layer
  – Currently TCP
  – Possible: RS232, JTAG, USB etc
• Channels transmit Messages
  – Asynchronous command / response with tokens
  – JSON marshalling
  – Events – order guaranteed
  – Progress
  – Can Proxy / Tunnel channels through value-add
• New: Streams service on top of Channels
What's in the Box

• Available from the git Repository:
  – Lightweight configurable plain C agent
    • Compiles out of the box on Linux, Windows native, Cygwin
    • Easy to port to other OS, e.g. VxWorks, Symbian
    • Supports basic debugging, file transfer, proxy and auto-discovery
  – Plain Java protocol framework org.eclipse.tm.tcf.core
  – Eclipse layer org.eclipse.tm.tcf for lazy loading (extension reg.)
    • Exemplary RSE Plug-in and Platform/Debug integration
  – A couple of exemplary plain C commandline utilities
    • New: plain C Debug Server value-add
  – Protocol specification, Getting Started documentation

• [http://wiki.eclipse.org/TCF](http://wiki.eclipse.org/TCF) and last year's TCF Tutorial
Demo

• ./agent -L-
• Connect RSE
  – discovers agent
  – File transfer, processes
• Build CDT Sample app
• Launch Debugger
Current Status of Adoption

• Stable core protocol specification
  – Standardization at Power.org (hardware connectivity)
  – Working on Services standardization

• Stable C agent framework
  – Initial use in commercial products, e.g. Wind River Workbench
  – Open Source use in EDC (Eclipse Debugger for C/C++), Nokia
  – Whitebox Adoption Model: Freescale, Atmel, Mentor
    • Migrate proprietary legacy agents into TCF agent plugins
    • TCF value-add for conversion from legacy protocol

• Exemplary RSE and Platform/Debug Integration

• Support for Tracing (Ericsson, Linuxtools, Polymtl.ca)
What's New in Helios

• Streaming and Zero-Copy Binary Transfer (LTTng)
• Agent: Dynamic Loading of Services as Sharedlib
• Debug Server Value-Add
  – Splitting ELF reading from agent
• SSL connection to the agent
• Formalized a Programming Pattern for guaranteed data consistency when dealing with multiple data sources
  – ACPM (Asynchronous Cache Programming Model)
  – Used in the debug client and TCF Server value-add
  – Asynchronous implementation of File System Service
References / Q&A

• [http://wiki.eclipse.org/TCF](http://wiki.eclipse.org/TCF)
  – Overview, Getting Started Docs
  – Code Repository access, Mailing list access
• EclipseCon 2009 TCF Tutorial

Questions ?