EDC Design Overview

Eclipse Debugger for C/C++

A Complete Architecture for C/C++ Debugging in Eclipse

- Leverages and connects existing Eclipse debug technology (Platform, CDT, DSF, TCF)
- Unlike current CDT debuggers EDC doesn't require a debug "back-end"
- Completely asynchronous for best performance
- Pervasive multi core/context/process support
- Provides a collection of core debug services
- Uses platform specific low level debug agents
- Reference implementations
  - Windows
  - Linux
  - Symbian OS

Eclipse Debugger Projects

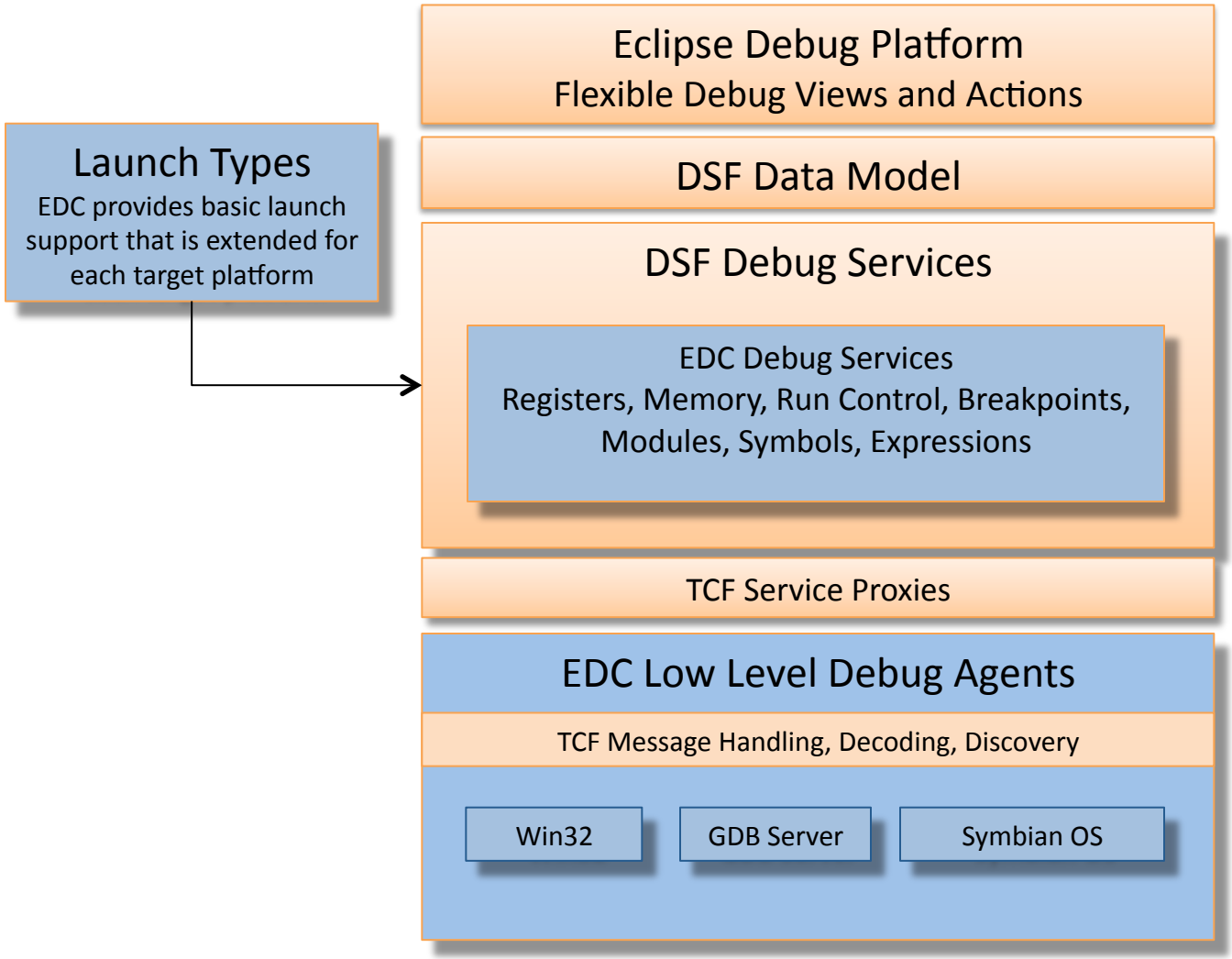
EDC is not a new debug architecture but instead leverages and extends work from these three Eclipse projects to provide a more complete set of debug components.

- Eclipse Debug Platform
  - Common generic debug UI and basic services
  - Flexible View Model
- Eclipse Debug Services Framework (DSF)
  - Services that provide data to the flexible view model
  - Support for most C/C++ debug elements
  - A component of the C/C++ Development Tools (CDT) project
- Eclipse Target Communications Framework (TCF)
  - Simple, flexible, extensible protocol for host to target communications
  - Discovery method for clients to find services on a target device

No “back-end” needed

- Existing Eclipse/CDT debuggers are required to integrate with a “back-end” that provides additional debug support
- EDC removes this requirement by providing core debug support and using low level debug agents for target specific services
- Example: EDC’s breakpoint service handles resolving the source location for a breakpoint into one or more locations in a module of code. It then calls target platform specific code that actually sets the breakpoint.

Architectural Overview



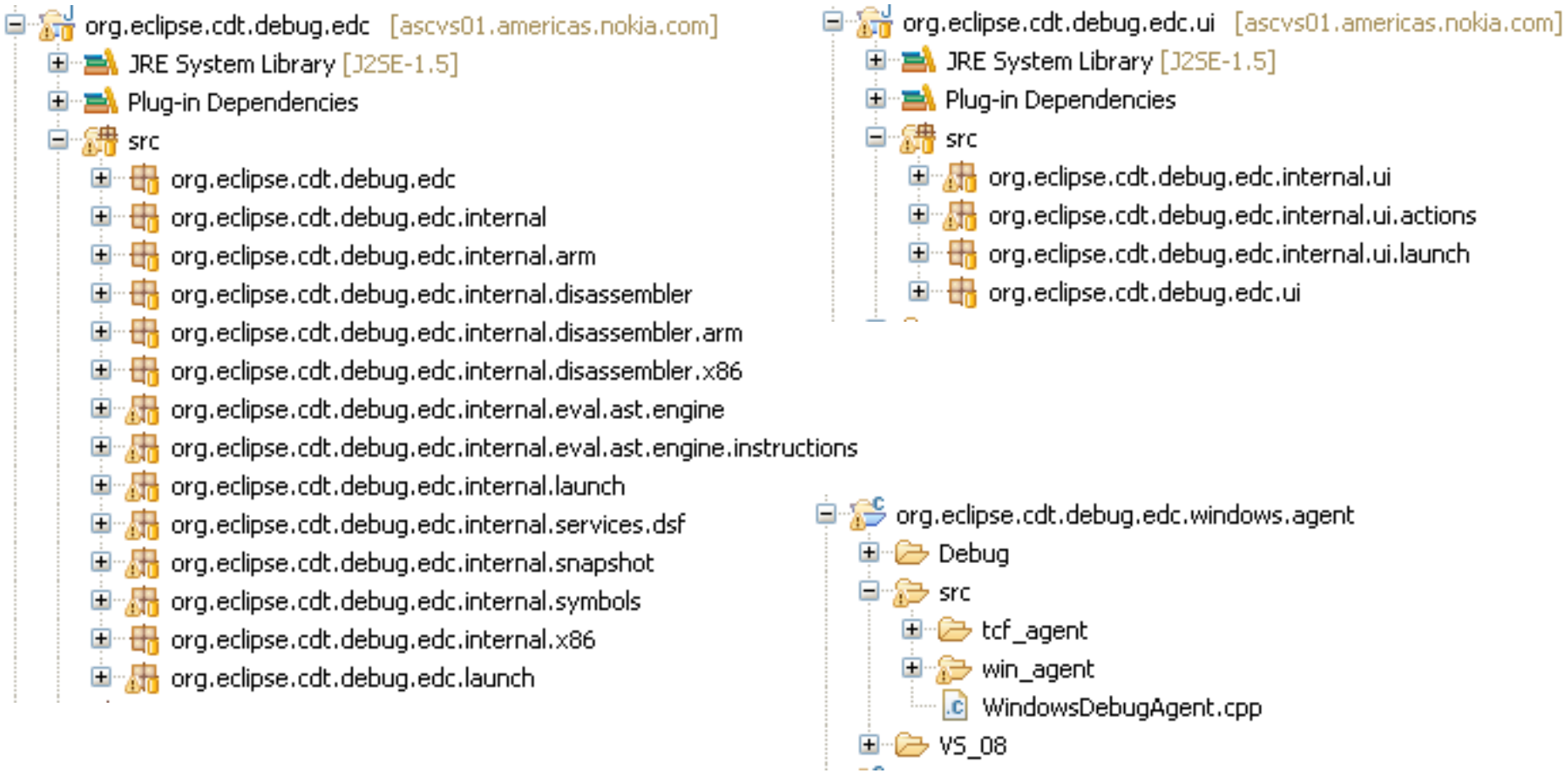
Inside an EDC Debugger

- A debugger for a specific target platform is a collection of debug services and agents
- When a debug session is launched it creates the services and connects to the agents required for that target platform
- Support for a target platform is built from some core EDC components and some created or customized for the target
- Agents can be written in Java or C/C++, can run on the host or remotely

Example: Inside the EDC Windows Debugger

- A Windows launch delegate creates a set of DSF Services
  - Common services: Memory, Run Control, Breakpoints, Modules, Expressions
  - Custom services: Windows Stack, x86 Disassembly Service
- And connects to the Windows Debug Agent
  - Windows debug agent uses the Win32 debug API to start and terminate processes, read and write memory and registers, and monitor debug events (exceptions, breakpoints)

EDC Plug-ins


org.eclipse.cdt.debug.edc [ascvs01.americas.nokia.com]
JRE System Library [J2SE-1.5]
Plug-in Dependencies
src
org.eclipse.cdt.debug.edc
org.eclipse.cdt.debug.edc.internal
org.eclipse.cdt.debug.edc.internal.arm
org.eclipse.cdt.debug.edc.internal.disassembler
org.eclipse.cdt.debug.edc.internal.disassembler.arm
org.eclipse.cdt.debug.edc.internal.disassembler.x86
org.eclipse.cdt.debug.edc.internal.eval.ast.engine
org.eclipse.cdt.debug.edc.internal.eval.ast.engine.instructions
org.eclipse.cdt.debug.edc.internal.launch
org.eclipse.cdt.debug.edc.internal.services.dsf
org.eclipse.cdt.debug.edc.internal.snapshot
org.eclipse.cdt.debug.edc.internal.symbols
org.eclipse.cdt.debug.edc.internal.x86
org.eclipse.cdt.debug.edc.launch
org.eclipse.cdt.debug.edc.ui [ascvs01.americas.nokia.com]
JRE System Library [J2SE-1.5]
Plug-in Dependencies
src
org.eclipse.cdt.debug.edc.internal.ui
org.eclipse.cdt.debug.edc.internal.ui.actions
org.eclipse.cdt.debug.edc.internal.ui.launch
org.eclipse.cdt.debug.edc.ui
org.eclipse.cdt.debug.edc.windows.agent
Debug
src
tcf_agent
win_agent
WindowsDebugAgent.cpp
VS_08