Tools for Fast Integration of DaVinci Multimedia Codecs into Applications

SDO Applications Team
Agenda

- **Introduction / Background**
  - How we got here; Which versions of CE, XDC to use…

- **QualiTI revival**
  - Checking XDAIS compliance; Top10 codec recommendations…

- **Auto-generating codec packages**
  - RTSC codec packaging wizard…

- **Auto-generating DSP executables**
  - RTSC server wizard, creating combos, leveraging configuro…

- **The Digital Video Test Bench (DVTB)**
  - Standard app for evaluating VISA codecs, scripting to ensure correct params

- **One-page Applications (DMAI)**
  - Simple apps you can write that work on dm6437, 6446, 6467, 355 etc.

- **Combos R Us**
  - Pre-cooked combos you can pick up off the shelf…

- **Conclusions**
How we got here…

• Lots of FAEs expressing dissatisfaction – captured here [link]
  – Difficulties understanding which versions of CE, XDC to use on which chips
  – Codec Engine 1.x usability concerns (issues with package compatibility/dependencies, memory map mismatches with Link etc)
  – Several codecs initially with ‘bad packaging’ making integration tricky
  – Lack of tools initially to help with XDAIS compliance, packaging
  – Difficulty comprehending ‘what next?’ when hitting XDC errors
  – Debugging a complex system

• Worldwide FAE tour to address CE, XDC, DVSDK Ease of Use concerns
  – Demonstrate new tools, new Ease of Use features to speed up integration
  – Show how to go from an MPEG4 codec library to a full DVSDK Codec Engine video application in about an hour
  – Hands-on lab sessions to gain familiarity with new flow
  – Reusable slide set for ASPs, customers – everything posted externally
Starting with the right CE, XDC, FC…

NOTICE

WHICH TOOLS AND CONTENT DO I START WITH?

WHAT VERSIONS ARE REQUIRED?
CE, XDC version matrix…

• Tip #1: use at least CE 1.20.02 and XDC 2.95.01!
  – Compatibility / dependencies issue with versions prior to this. See this [link] for full details.

• Tip #2: baseline from >= DVSDK 1.30 or 1.40
  – Dm6467 requires >= DVSDK 1.40 (supports dm6467)
  – Dm355 requires >= DVSDK 1.30 (supports dm355, dm6446)
  – Dm6446 does not require either of these – but DVSDK 1.30 contains CodecEng 2.x, XDC 3.x – major usability improvements – highly recommended.

• Tip #3: download full CE distro with ‘cetools’ subdir [link]
  – Codec Eng comes in 2 flavors: with & without cetools. Details here [link]
  – DVSDK ships with CE without cetools
  – Downloading ‘full’ CE (a) guarantees you have all the right FC, DSPLink, CMEM, XDAIS content (b) makes setting XDCPATH much easier.
Will this codec play well with others?
**Checking XDAIS compliance**

- **The broken window concept**
  - If there’s no police force eventually there will be some broken windows!

- **Previous QualiTI tool fell into disrepair**
  - Didn’t keep up with new ISAs
  - Didn’t keep up with new challenges – e.g. guidelines for how to access accelerators, DMA in ‘co-operative’ fashion

- **New QualiTI release…**
  - Released Mar ’08 in XDAIS 6.10 [link]
  - Truly excellent Ease Of Use
    - For each rule fail indicates why, possible causes, and probable fixes
  - Tests the key rules that prevent *integration* problems
QualiTI Look & Feel

- GUI & command line support
- Produces HTML report to ship with algorithm package
- Runs various object file analysis tools to check rules e.g. namespace compliance via Codegen ‘nm6x’ tool
- External wiki topic here [link]
Deepdive on a few rules...(1)

- Rules 8, 9, 10: Namespace compliance

  - ... Anything with FIR_TI_ prefix is good, _memcpy is ok since it comes from allowed RTS functions list. _dummy1 is bad. This is not namespace compliant.
Deepdive on a few rules...(2)

- Rule 13+: correct linker section names

  ![Image of a table showing a rule's status and description](image)

  **Description**
  
  Each of the IALG methods implemented by an algorithm must be independently relocatable.

  Additional rule: there should be no non-standard linker sections. (Standard linker sections are: ".cint", ".switch", ".far", ".text", ".const", ".bss", ".pint").

  **Status Details (failure 2 of 3)**
  
  Test failed. There is a non-standard linker section: ".tables".

  File: "fir_t1_filter.064".

  Non-standard linker sections are dangerous because the linker can place them in the wrong type of memory. To fix this problem, rename your non-standard section to be a

- Observe ".tables" section. Is this code, uninitialized data or const data?
  - Linker may place this in wrong type of memory (eg L1PSRAM) if not explicitly placed
  - Frequent source of crashes

- How to fix this?
  - Rename to be subsections of main compiler sections e.g. ".const:tables"
Additional Codec Producer Guidelines

• Codec ‘Top 10 recommendations’
  – Published on external wiki [here]
  – Based on lessons learned from previous customer integration problems
  – Examples: -
    • Perils of extending VISA XDM classes the ‘wrong’ way (inArgs, outArgs etc)
    • Publish ‘known good’ parameters & explicitly detail compatibility breaks
    • Performance benchmarks in context of a "real system" e.g. a Codec Engine based app such as DVTB
FAQs

• Q: Will you mandate XDAIS compliance?
  • A: The HTML report produced by QualiTI will be a prerequisite for ASP / TI deliveries going forward

• Q: Is QualiTI in the DVSDK?
  • A: Yes, its in DVSDK 1.40 (XDAIS 6.10)

• Q: Will you extend the # rules tested?
  • A: Yes. Looking at ‘dynamic’ options i.e. using new standard DSS loadti [link] loader to run test cases
    – E.g. test for incorrect scratch memory usage, check for DMA resource hard-coding
QualiTI Demonstration
Auto-generating Codec Packages

What kind of box would you like my codec in?
Codec Packages, Servers, Engines...

Key Points

- Each codec in separate RTSC package
- Server package == DSP executable. Servers applicable only in GPP+DSP environment
- Engines invoke servers to instantiate remote codecs
- Can have multiple engines per application
- Wizards available to automate creation of codec & server packages. DVTB can be easily reconfigured for any server

Codec Packages, Servers, Engines...
Value of RTSC codec packaging

• Standard ‘box’
  – No need to specify different –I"/codecPath1" – 
  I"/somethingElse(codecPath2" preprocesser directives. Can have 1 
  package path for multiple codecs

  – Codec-Engine add-on methods (e.g. getStackSize() ) enable 
  specification of stack size for optimal CE Task stack creation.

  – Linker template allows contribution of required placement directives on 
  per-codec basis. No messy “cut-paste from this example link.cmd”

  – Enables additional tooling e.g. repoman Repository Manager

  – Learn one & you’ve learned them all.

• Can still use package in non-XDC environment
  – Files, functions are just additions – non-XDC client can just point to –I list
  and library “same as it ever was”
Who Creates What?

A Codec Producer with a Codec library…

Creates a Codec Package

And then creates a Server Package

That can be tested using DVTB

Which is then easily consumed by a Codec Consumer
Automating via RTSC Package wizard

Library File
Docs
Test App
Release Notes
Header Files

RTSC
Codec
Package Wizard

Basic Package Info:
Package Name
Target
Codec Class
Version Number

All necessary XDC Files in the correct directory structure
RTSC Package wizard look & feel (1)

List of codec classes derived from CE ICodec – hence always up to date (VIDDDEC2 etc)

Optionally create CE ‘add-on’ files/methods
Using tools (nm6x, OFD, cg_xml [link] etc) automatically retrieves IALG, IDMA, IRES symbols.

Auto-calculates worst case stack-size – can be overriden

Cannot calculate scratch memory sizes
  – Would need to execute the algorithm in a test harness for this
  – User can fill these in based on codec datasheet
RTSC Codec Package Wizard: IS / IS NOT

- **IS**
  - Only for codecs with interfaces inheriting from ICodec
    - E.g. IVIDDEC, IAUDENC, IVIDANALYTICS
- **DOES**
  - Generate basic XDC files for codecs
- **CAN**
  - Determine stack size requirements*
  - Use information contained within the codec library to determine global variable and section information*

- **IS NOT**
  - A check for xDAIS compliance
- **DOES NOT**
  - Execute “xdc release” or generate release tar files
- **CANNOT**
  - Determine all inputs needed based on the library
    - E.g. scratch size

*For libraries created with TI Code Generation Tools
Where to get the RTSC package wizard(s)?

• Available: -
  – On external web [link]
  – In DVSDK 1.40

• Contains:
  – RTSC Codec Package Wizard
  – RTSC Server Package Wizard
  – Documentation
    • Installation and Use Instructions, FAQ
    • External wiki page [link] with constant updates

• Command line and GUI options for both Windows and Linux
FAQs

• Q: Does the wizard work for dm355 gcc codecs?
  • A: Yes. It doesn’t auto-calculate stack sizes or place sections but it still generates the right ‘box’.

• Q: Are all TI codecs generated (initially) via this tool?
  • A: Yes, from 4Q07

• Q: Do Davinci Application Service Providers (ASPs) know about this tool?
  • A: Yes. 2 webexes held in 2H07 + TIDC session + external wiki + marketing.

• Q: Anything special for DM6467?
  • A: From codec perspective no. Server/combos have additional config for accelerator setup (which the server wizard generates)

• Q: what’s the minimum version of XDC tools & CE the wizard needs?
  • A: XDC tools 2.95 and Codec Engine 1.20.02.
RTSC Package Wizard Demonstration
Auto-generating DSP Executables
Quick Terminology Recap

• Codec Package
  – RTSC package containing a codec library
  – May integrate into Codec Engine

• Server Package
  – Integrates codecs, framework components, and system code
  – Servers only exist in GPP+DSP configurations!
    • “Engine” invokes codec servers to instantiate \textit{remote} codecs
  – Unlike codecs, servers are on a per platform basis
    • Memory maps are different between DM6446 and DM6467
  – GPP uses VISA APIs to transparently invoke remote codecs on the DSP
  – Codec Combos vs. Unit Servers:
    • A “Unit Server” is a DSP Executable containing exactly 1 codec
    • A “Codec Combo” is a DSP Executable containing > 1 codec
Automating via RTSC server wizard

Codec Package
- Must be pre-built
- Format should be consistent with the RTSC Codec Package Wizard format

RTSC Server Wizard

Basic Package Info:
- Package Name
- Platform

All necessary Files to create a server!
RTSC server wizard look & feel

Codec Package Name

Default Output Repository is the Codec Package’s Repository! Change the Output Repository by selecting this button!

Command Line Version is also available! Run: xs ti.sdo.codecutils.genserver --help

Notice this Codec Package has module MPEG2DEC

Codec Package Name: mycompany.mygroup.codecs.mpeg2dec
Module Name: MPEG2DEC
Platform: ti.platforms.evmDM5446
Server Package Name: mycompany.mygroup.servers.mpeg2dec_unittests

Set Output Server repository: C:/demo
Role: Contains the applications main() function. Notice:

1. Initialize Codec Engine
2. Initialize Trace

DSP/BIOS runs its threads AFTER main() completes, so main() should only contain initialization statements
Deepdive on some generated files (2)

```javascript
/* ======== codec.cfg ========*/

var MPEG2DEC = xdc.useModule('mycompany.mygroup.codecs.mpeg2dec.ce.MPEG2DEC);

    // Package Config
    MPEG2DEC.alg.watermark = false;
    MPEG2DEC.alg.codeSection = "DDR2";       // don't forget the quotes!
    MPEG2DEC.alg.udataSection = "DDR2";
    MPEG2DEC.alg.dataSection = "DDR2";

Server.algs = [

    {name: "mpeg2dec", mod: MPEG2DEC, threadAttrs: {
        stackMemId: 0, priority: Server.MINPRI + 2, groupIdx : 0,
    }}
];
```

- **Role:** Specify codec package to use and its configurable parameters
- **This is the 1 file you **MUST** edit – wizard cant know where you want to place such sections.**
- **The sections must be placed in a memory section defined in server.tcf!**
- **Tuning:** modify server.cfg, server.tcf to optimize DMA, memory resource utilization.
RTSC Server Package wizard IS / ISNOT

- **IS**
  - Only for codecs with interfaces inheriting from ICodec
    - E.g. IVIDDEC, IAUDENC, IVIDANALYTICS
- **DOES**
  - Generate server package files for a single codec server
    - E.g. aacdec_unitserver_evmdm6446.x64P
- **CAN**
  - Determine config params from the codec package
    - E.g. if the watermark Boolean’s default value is set to false in the codec package, then the server configuration file (codec.cfg) will show the same value

- **IS NOT**
  - A check for XDAIS compliance

- **DOES NOT**
  - Execute “xdc release” or generate release tar files
  - Automatically place sections in memory
  - Provide memory maps other than for DVSDK
  - Create Server Combos
    - But the unit test servers are a good place to start from when creating combos [link]
FAQs

• Q: Does the server wizard accept dm355 gcc codecs?
  • A: No. Server concept non-existent for Arm-only (or DSP-only) platforms

• Q: What's the value in a unit-server? I need a combo!
  • A: Unit-server enables:
    – Rapid eval of codec quality in real-world Codec Engine DVSDK environment
    – Confirmation that 1st phase integration (OS, framework etc) is OK.
    – Rapid demos for the Field, Sales Force

• Q: Can I create a combo with this tool?
  • A: Not directly. However it's simple to extend by hand – see external wiki FAQ [link]

• Q: How much config / tuning do I need to do?
  • A: It varies. Codec producer likely just sets memory section placement. System combo integrator likely tweaks DMAN3, DSKT2 parameters to tune memory & resources usage.
RTSC Server Wizard Demonstration
Using Standard Build Flows with RTSC Packages
Building servers with standard makefiles

• Content generated by wizard leverages XDC build
  – ‘should just work’
  – however standard makefiles or CCS pjt files can also be used instead.

• Configuro enables RTSC config in user’s own build flow
  – Modeled after TCONF utility of DSP/BIOS 5.X
  – Processes an RTSC config script into a set of header files, object and library files, and compiler/linker command line options
  – User refers to these generated files in their own build flow
  – Tool chain agnostic
    • Works with any compiler supported in XDC, e.g. TI, GCC, VC++
    • Works with any build flow, e.g. CCS, make
Using Configuro from Make

• One extra rule
  – Runs configuro command
  – Depends on .cfg file
• Options to describe h/w
  – -t target (ISA and tool chain)
  – -p platform (board)
• Compiler invocations
  – Take switches from generated “compiler.opt”
• Linker invocations
  – Include generated “linker.cmd”
• Configuro always runs
  – Adds ~1 second if no change
  – Triggers relink or recompile if change, depending on change

CGTOOLS = c:/CCStudio_v3.3/C6000/cgtools
CC = $(CGTOOLS)/bin/cl6x $(CFLAGS)
RTS = $(CGTOOLS)/lib/rts6400.lib
XDCROOT = c:/xdc_3_00_00
XDCPATHF = c:/myrepo/xdcpath

all: hello.out
## run configuro
%/linker.cmd %/compiler.opt : %.cfg
$(XDCROOT)/xs @$(XDCPATHF) xdc.tools.configuro
\
-c $(CGTOOLS) -t ti.targets.C64 \
-p ti.platforms.sim6xxx $<

## compile user’s C files
%.obj : %.c mycfg/compiler.opt
$(CC) -@mycfg/compiler.opt -c $<

## link user’s executable
hello.out : hello.obj mycfg/linker.cmd
$(CC) -z -o @ -c $^ $(RTS)
Using Configuro from CCS

- XDC installs into CCS 3.3
- Builds .cfg using configuro
- Choosing an XDC version is same as choosing a DSP/BIOS version
- New XDC tab in CCS build options
  - Same fields available as in configuro command line
  - Active if project has .cfg file
- Works with TI code gen for C6000 and ARM ISAs
  - No C5000, C2000, GCC
- Double-clickable error messages
  - Opens script in editor at line
Codec Engine video_copy Server Example

• Requires “--tcf” flag
  – Enables .tcf file in config step as well as .cfg
  – Limitation: need same filename prefix e.g. app.tcf paired with app.cfg

• Needed for DSP/BIOS 5
  – TCF file is of the same format as without configurō

```
# [CE] The generated config package depends on the 2 config scripts. $(CONFIGPKG) : $(CONFIGPKG).cfg
   $(CONFIGPKG).tcf
       $(CONFIGURO) -c $(CGTOOLS) -o $(CONFIGPKG) \
          -t $(XDCTARGET) -p $(XDCPLATFORM) \n           --tcf $(CONFIGPKG).cfg

# "normal" makefile settings and rules follow.
%.obj : %.c $(COMPILER_FILE)
    $(CC) $(CFLAGS) -c $<

# app.out, in addition to its standard stuff, must specify the auto-generated $(LINKER_FILE)
$(SERVER_EXE): main.obj $(LINKER_FILE)
    $(LINK) -o $@ -c $^`
...
The Digital Video Test Bench (DVTB)
DVTB Introduction

- Digital Video Test Bench (DVTB)
  - Provides an easy interface to rapidly test codec servers
  - Available in the DVSDK
  - Supports encode/decode of all VISA classes
  - Supports configuration of audio & video peripherals
  - Scriptable

```bash
# Specify the codec combo to be used
setp engine name mpeg4dec_engine

# Specify the decoder to be used
setp viddec codec mpeg4dec

# Specify number of frames to decode
setp viddec numFrames 1000

# Trigger the decode+display scenario
func viddec -s /opt/dvevm/data/videos/davincieffect_ntsc.mpeg4
```
Using DVTB to test codec servers

• One application, many servers
  – Engine.createFromServer (new in CE 2.00)
    • Essential new Ease Of Use feature
  – Configure entire ARM application’s engine with a single line of code
    • No need to load codec modules – avoids duplication
    • No need to set up DSP memory map info – avoids mismatch headaches!
  – Each server has an auto-generated “Server info file” used by this API

```c
/*!*
 * ======= dvtb-codecs-6446.cfg =======
 */
...
var Engine = xdc.useModule('ti.sdo.ce.Engine');
Engine.createFromServer(
  "g711dec_engine", // engine name
  "./g711dec_unitserver_evm6446.x64P", // server executable
  "ti.sdo.servers.g711dec_unitserver_evm6446" // server path
);

Engine.createFromServer(
  "mpeg4dec_engine", // engine name
  "./mpeg4dec_unitserver_evm6446.x64P", // server executable
  "ti.sdo.servers.mpeg4dec_unitserver_evm6446"
);
```
**App config – before & after CE 2.x**

### Codec Engine 1.20

```javascript
/* Encoders */
var H264VENC =
    xdc.useModule('ti.sdo.codecs.h264venc.ce.H264VENC');
var MP4VENC =
    xdc.useModule('ti.sdo.codecs.mp4venc.ce.MP4VENC');
var G711ENC =
    xdc.useModule('ti.sdo.codecs.g711enc.ce.G711ENC');

/* ==== Engine Configuration ==== */
var Engine = xdc.useModule('ti.sdo.ce.Engine');
var videncode_engine = Engine.create("videncode_engine", [
    {name: "mp4venc", mod: MP4VENC, local: false},
    {name: "h264venc", mod: H264VENC, local: false},
    {name: "g711enc", mod: G711ENC, local: false},
  ]); 
videncode_engine.server = "/./videncode_combo_evmdm6446.x64P";
```

### Codec Engine 2.x

```javascript
/* ===== Engine Configuration ===== */
var Engine = xdc.useModule('ti.sdo.ce.Engine');
Engine.createFromServer( 
    " videncode_engine ",
    "/./videncode_combo_evmdm6446.x64P ",
    "ti.sdo.servers.videncode_combo_evmdm6446"
);
```

- A lot less typing! (equals a lot less errors 😊)
- `createFromServer` shows how DVTB (or any application e.g. demos) finds the server and configures Codec Engine to use it.

---

Texas Instruments

43
DVTB Sample results

root@<ipaddr>:/opt/ce2x_xdc3x_wizard_fsys# ./dvtb -s dvtb-scripts/mpeg4-play.dvs
setp engine name mpeg4dec_engine
   PASS: setp

# Specify the video decoder to be used
setp viddec codec mpeg4dec
   PASS: setp

# Trigger the mpeg4 decode+display scenario
func viddec -s /opt/dvevm/data/videos/davincieffect_ntsc.m2v
[DVEVM-ST-LOG]: dvevmStFuncDispatcher.c: Starting Thread for command <viddec>
[DVEVM-ST-LOG]: dvevmStVidPlay.c: Video [Decode][Play] started
[DVEVM-ST-LOG]: dvevmStVidPlay.c: VPBE device </dev/fb/3> initialized for display
[DVEVM-ST-LOG]: dvevmStEngine.c: Engine <mpeg4dec_engine> opened
[DVEVM-ST-LOG]: dvevmStVidPlay.c: Video Decoder <mpeg4dec> initialized
[DVEVM-ST-LOG]: dvevmStVidPlay.c: Frame#, 1, Decoded Frame Size(bytes), 7517, DecTime(us), 10208
[DVEVM-ST-LOG]: dvevmStVidPlay.c: Frame#, 2, Decoded Frame Size(bytes), 25025, DecTime(us), 8839

• Video output on monitor
• Read & write codec parameters via ‘getp’ and ‘setp’ as necessary in .dvs script or via command line
• For CE 2.x users: See more debug info via: -
   – CE_DEBUG=2 ./dvtb -s dvtb-scripts/mpeg4-play.dvs
FAQs

• Q: Does DVTB work for dm355?
  A: Yes. Available for all Davinci platforms (dm6437, 355, 6446, 6467 etc)

• Q: Does this work with custom combos?
  A: Yes

• Q: Can I run lots of Engines consecutively (scripting)?
  A: Yes.

• Q: Do I have to use XDC build for this?
  A: No. DVTB uses standard Makefiles and runs configuro as part of that flow.

• Q: Is this in the DVSDK?
  A: Yes. Its in all DVSDKs
One Page Applications (DMAI)

I want to write my own app. Got anything simpler than DVTB?
DMAI Motivation

- DVTB is an excellent platform for video quality analysis, scripting, combo evaluation.
  - However it is a complex application
- Also need simpler application entry point – ideally “1-page” apps
- Davinci Multimedia Application Interface (DMAI) sits here:
DMAI Is:

- A thin utility layer
  - Sits on top of the operating system (Linux or DSP/BIOS) and Codec Engine (CE).

- Driven by the DVSDK demo applications, which need to be ported quickly to new platforms.
  - DMAI is present in DVSDK 1.40 and later.

- A collection of modules, abstracting a common peripheral or codec operation (Frame copy, Audio decode, etc.).
  - Information is passed between modules using a Buffer abstraction.
Dmai_init();

/* Module and buffer to manage input from “myfile.aac” */
Loader_Handle hLoader = Loader_create("myfile.aac", &lAttrs);
Buffer_Handle hInBuf;

/* Module and buffer to manage output to sound peripheral */
Sound_Handle hSound = Sound_create(&sAttrs);
Buffer_Handle hOutBuf = Buffer_create(Adec_getOutBufSize(hAd),
   &bAttrs);

/* Module to manage audio decode */
Engine_Handle hEngine = Engine_open("myengine", NULL, NULL);
Adec_Handle hAd = Adec_create(hEngine, "aacdec", &params,
   &dynParams);

/* main algorithm */
Loader_prime(hLoader, &hInBuf);
while (1) {
   Adec_process(hAd, hInBuf, hOutBuf);
   Sound_write(hSound, hOutBuf);
   Loader_getFrame(hLoader, hInBuf);
   if (Buffer_getUserPtr(hInBuf) == NULL) break;
}
FAQs

• Q: Does DMAI work for dm355?
  A: Yes. Will be available for all Davinci/OMAP platforms (dm6437, 355, 6446, 6467 etc)

• Q: Do I have to use XDC build for this?
  A: No. DMAI uses standard Makefiles and runs configuro as part of that flow.

• Q: Is this in the DVSDK?
  A: Yes. Its in DVSDK >= 1.40. The demos (encode, decode, encodedecode) all leverage DMAI.

• Q: Is source code provided?
  A: Yes, all of it.

• Q: Can I reuse the same application code on DSP-only platforms?
  A: Yes, this is one of the key benefits of DMAI. However not all modules are ported across all platforms e.g. peripherals not validated on DSP-side of DM6446 hence no drivers.

• Q: Does DVTB leverage any of the DMAI modules?
  A: No (as of 1Q08). However evaluation is under-way…
Pre-fabricated Combos

Coming Soon...
Combos R Us !!
Combos R Us background / concept

• Background
  – If customer needs a codec that’s outside of the DVSDK requirements, the commissioning process is an obstacle
    • Amount of testing on non-DVSDK combos may be less
  – Don’t need 10s of people generating ‘standard combos’

• Concept
  – Repository / store for FAE-requested combos.
    • Web front-end: select codecs, platform from drop-down lists
    • Uses only evaluation codecs (watermarked etc)
      – Sends email to Catalog marketing for tracking purposes.
    • Points you to pre-generated, pre-tested combo
      – Plus example .dvs file to run in DVTB
    • Enables “pick up & demo” for Sales Force
    • Field contributions welcome – TI-internal open-source

• Coming soon…!
Summary

• Presentation + labs designed to help DVSDK Ease of Use

• Several new tools & upgrades available to improve usability: -
  – RTSC codec package & server wizards
  – Codec Engine 2.x features
  – Configuro : use standard makefiles or CCS project files
  – Improved error messages in XDC 3.x

• DVTB and DMAI enable rapid application prototyping

• Pre-fabricated combos will further improve Davinci content deployment
Resources

• Downloads
  – RTSC : [link]
  – XDAIS : [link]
  – Codec Engine : [link]
  – RTSC Package Wizards : [link]

• Documentation
  – RTSC
    • Open-source Eclipse wiki : [link]
    • XDC Getting Started Guide (available in /XDC_x_xx/docs)
    • XDC User Guide
  – XDAIS
    • QualiTI : [link]
  – RTSC Package Wizards Wiki Page : [link]
  – TI eXpressDSP Wiki : [link]
  – Codec Engine
    • Codec Engine Wiki Page [link]
    • Codec Engine Server Integrator’s Guide (SPRUED5A)
    • Codec Engine Algorithm Creator User’s Guide (SPRUED6B)
  – DVTB
    • DVTB User Guide (available in /DVSDK_x_xx/dvtb_vx_yy_zz/docs)