



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

Power.org™ Standard for Common Debug Interface API

APPROVED Version 1.0 – 18 September 2008

Copyright © 2008 Power.org. All rights reserved.

The Power Architecture® and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

Implementation of certain elements of this document may require licenses under third party intellectual property rights, including without limitation, patent rights. Power.org and its Members are not, and shall not be held, responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

THIS POWER.ORG SPECIFICATION PROVIDED “AS IS” AND WITHOUT ANY WARRANTY OF ANY KIND, INCLUDING, WITHOUT LIMITATION, ANY EXPRESS OR IMPLIED WARRANTY OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL POWER.ORG OR ANY MEMBER OF POWER.ORG BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, EXEMPLARY, PUNITIVE, OR CONSEQUENTIAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Questions pertaining to this document, or the terms or conditions of its provision, should be addressed to:

IEEE-ISTO
445 Hoes Lane
Piscataway, NJ 08854
Attn: Power.org Board Secretary

1 Introduction

2 Power.org's mission is to develop, enable, and promote Power Architecture® technology as the preferred
3 open standard hardware development platform for the electronics industry and to administer qualification
4 programs that optimize interoperability and accelerate innovation for a positive user experience. Power.org
5 seeks to solicit the participation of all interested parties on a fair, equitable, and open basis.

6 Power.org's output includes:

- 7 • Open standards and specifications
- 8 • Business guidelines documents
- 9 • Best practices and education
- 10 • Certifications to validate implementations and drive adoption

11 Power.org's specifications will enable:

- 12 • Interoperability between community members
- 13 • Sustainability built on driving open standards and convergence

14

1 **Revision History**

2

Version	Date	Editor	Description
1.0 D1	06 May 2008	Steve Furr	Preliminary Draft Specification
1.0 D2	22 May 2008	Joan Woolery	Grammar and cosmetic changes by tech writer (Madeleine)
1.0 D3	23 May 2008	Steve Furr	Included definition of normative terminology
1.0	18 September 2008	Joan Woolery	Power.org Board approved the draft spec. Removed “DRAFT” from header and front matter.

3

4

1 **Table of Contents**

2 Introduction2

3 Revision History3

4 1 Overview5

5 1.1 Scope5

6 1.2 Purpose5

7 1.3 Terminology5

8 1.4 Abbreviations and Acronyms6

9 2 Technical Content.....6

10 2.1 Services Framework7

11 2.2 Standard Services8

12

1 Power.org™ Standard for Common Debug 2 Interface API

3 1 Overview

4 1.1 Scope

5 The Common Debug Interface API specification seeks to provide a common method for communicating
6 with target systems. It covers a protocol specification as well as an application programming interface
7 (API) for accessing the protocol. The protocol specification provides an extensible mechanism for
8 accessing different services made available over the protocol. This standard also covers a set of normative
9 services for the protocol specification and Java™ language bindings to access the services.

10 The Common Debug Interface API is structured as an Application Profile of the Target Communication
11 Framework (TCF) specifications.

12 1.2 Purpose

13 Today almost every device software development tool on the market has its own method of communicating
14 with target systems. Communication methods often conflict with each other, require individual setup,
15 configuration and maintenance, or impose all kinds of unnecessary limitations. The target communication
16 framework (TCF), referenced by this standard, is designed to establish common ground in the area of
17 communication protocols between development tools and embedded devices.

18 The primary purpose of this standard is to establish a standard protocol for defining and accessing new
19 target services. The standard additionally specifies a standard language binding for clients to locate and
20 access services and for target agents to deploy new services.

21 This standard is aimed at ensuring the suitability of the base specification in addressing the needs of
22 communicating with target agents that are:

- 23 • Embedded target agents for an application debugging under an operating system
- 24 • Hardware probes providing access to silicon targets using a debug port
- 25 • Debug agents for simulation environments

26 1.3 Terminology

27 In the context of this specification, the term *normative* is used to indicate that something is prescribed as
28 the standard. This is applied in particular to references to the underlying Target Communication
29 Framework specifications. In these cases, some elements of the TCF specifications may be described as
30 implementation or vendor specific. Any designation by this specification indicates that a particular

1 mechanism is required. For example, this specification cites that JavaScript Object Notation (JSON) is
2 normative as the data encoding standard for TCF services, meaning that an implementation must support it.

3 The normative term may be used in several contexts. When it is applied to references (e.g. to sections of
4 the underlying TCF specifications), it indicates that the specifications within that section must be followed
5 by any implementation conforming to this standard, except as specifically noted within this specification.
6 When applied to individual TCF services, an indication that the service is normative means that any
7 implementation of the TCF specification must conform to the definition of that service if *it* implements the
8 service. If a service must be provided by implementations conforming to this standard, it will be
9 additionally described as *mandatory*. When the term *normative* is applied to protocol specifications within
10 the underlying specification, or bindings to protocols, such as UDP or TCP/IP for transport layers, then it
11 means that a conforming implementation must support that protocol, but that implementations are free to
12 support alternative protocols as well.

13 1.4 Abbreviations and Acronyms

14 API Application Programming Interface

15 TCF Target Communications Framework

16 JSON JavaScript Object Notation

17 2 Technical Content

18 TCF consists of both a set of specifications for the protocol and a reference implementation in Java.
19 Additionally, there is a prototype TCF agent written in C.

20 Software implementations, including both the Java reference implementation and the prototype C services
21 agent, are not part of the standard.

22 TCF is hosted on the Eclipse.org website, under the Target Management subproject of the Device Software
23 Development Platform project. This standard incorporates Revision 113 of the TCF documentation as the
24 standard specifications.

25 The following table indicates which sections of the TCF documentation apply to this standard, and
26 indicates whether the reference is normative or informative.

27

28

Table 1 TCF References

Name	Title	Description	Status
TCF Project Overview	TCF Documentation, May 6, 2008	TCF project goals and results.	Informative
TCF: Getting Started	TCF Documentation, May 6, 2008	Getting started with TCF: creating Eclipse workspace, building agent, making a first connection	Informative

TCF Specifications	TCF Documentation, May 6, 2008	Design goals, requirements and format of TCF communication protocol, framework API and software design considerations.	Normative¹
TCF Services Definitions	TCF Documentation, May 6, 2008	TCF communication model is based on the idea of services. A service is a group of related commands, events and semantics. New services are expected to be defined by developers of tools and target agents. To achieve a certain level of compatibility between tools and targets, TCF includes definitions of some common services.	Normative²
TCF Context Identifier Explanation	TCF Documentation, May 6, 2008	Most if not all TCF services functions need some way to identify what entity, e.g., process, thread, task, or device on JTAG scan chain, they should operate on. To make this identification, TCF uses a context identifier (also called ContextId). This document explains how ContextIds are intended to be used.	Informative
TCF Agent Prototype	TCF Documentation, May 6, 2008	Brief description of the TCF target agent prototype implementation.	Informative

1

2 2.1 Services Framework

3 TCF provides a framework that implements a services-based model. This framework consists of a protocol
 4 for communications between clients and service managers (or target agents). The protocol consists of the
 5 means to construct requests of services and receive events and replies from services. Additionally, a Java
 6 language binding to the framework allows clients to discover services and make requests and allows new
 7 services to be deployed to target agents.

8 The TCF services framework is defined in the “TCF Specifications” reference. A number of sections of
 9 that reference are *informative*.

10 The “Overview” section of the reference material is *normative*. The “Requirements” subsection provides
 11 rationale for a number of the choices within the protocol.

¹ This section is normative, except as explicitly noted within this standard.

² This section is normative, except as explicitly noted within this standard.

1 The “Framework Software Design Considerations” section of the reference material is *informative*. The
2 “Message Ordering” sub-section describing requirements for the ordering of commands, events and
3 responses is *normative*.

4 The “Transport Layer” section of the reference material is *normative*.

5 The “Communication Protocol” section defining service commands, results, events and flow control over
6 messages is *normative*. The communication protocol covers the wire protocol for encapsulation of
7 messages and the representation for the marshaling of messages themselves. The data payload of messages
8 is covered by the separate data representation.

9 The “Preferred Marshalling” section of the reference material is *normative*. The JavaScript Object Notation
10 (JSON) shall be supported as the default data representation protocol for marshaling data within command,
11 event, or response messages.

12 The TCF framework does not generally specify required services. The single exception to this rule is the
13 Locator service. The Locator service is mandatory. It shall be provided by any target agent using the TCF
14 framework. The “Locator Service” section of the reference material is *normative*.

15 **2.2 Standard Services**

16 An implementer of a target agent is free to deploy whatever set of services he or she desires. The only
17 mandatory service is the Locator service.

18 To facilitate interoperability for run control operations, the TCF documentation specifies a common set of
19 service definitions. Not all of the service definitions are *normative* from the standpoint of this standard. All
20 of these service definitions are contained in the “TCF Service Definitions” normative reference.

21 Both the service protocol definition and the Java language binding to each service are *normative*. Only
22 service definitions explicitly cited in this standard are *normative*. All other service definitions are
23 *informative*.

24 The following set of services is *normative*:

- 25 • Run Control
- 26 • Memory
- 27 • Registers
- 28 • Breakpoints

29

1 **Annex A (Normative)**
2 **References**

3

4 The following references are normative to this specification:

5 TCF Target Communications Framework documentation, May 6, 2008, Eclipse.org,
6 http://dev.eclipse.org/viewsvn/index.cgi/org.eclipse.tm.tcf/?root=DSDP_SVN.

7

8