

TCG Trusted Network Connect

TNC IF-TNCCS: TLV Binding

Specification Version 2.0

Revision 10

23 January 2008

Draft

Contact:

Steve Hanna – shanna@juniper.net (Co-Editor, TNC Co-Chair)

Ravi Sahita – ravi.sahita@intel.com (Co-Editor)

Ryan M. Hurst – rmh@microsoft.com (Co-Editor)

Paul Sangster – Paul_Sangster@symantec.com (TNC Co-Chair)

Work In Progress

This document is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

TCG CONFIDENTIAL

TCG

Copyright © TCG 2005 - 2008

Copyright © 2005-2008 Trusted Computing Group, Incorporated.

Disclaimers, Notices, and License Terms

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

This document is copyrighted by Trusted Computing Group (TCG), and no license, express or implied, is granted herein other than as follows: You may not copy or reproduce the document or distribute it to others without written permission from TCG, except that you may freely do so for the purposes of (a) examining or implementing TCG specifications or (b) developing, testing, or promoting information technology standards and best practices, so long as you distribute the document with these disclaimers, notices, and license terms.

Contact the Trusted Computing Group at www.trustedcomputinggroup.org for information on specification licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.

Revision History

08/22/2007	2.0r0: First draft of use cases and requirements
08/30/2007	2.0r1: Revised use cases and requirements in response to feedback
9/5/2007	2.0r2: Added straw-man of protocol
9/13/2007	2.0r3: Updated protocol straw-man based on initial feedback
10/7/2007	2.0r4: Updated based on IF-TNCCS SDT feedback
10/29/2007	2.0r5: Updated based on TNC Oct 9-10 F2F
11/7/2007	2.0r6: Updated based on TNC Oct 30 Virtual F2F
11/20/2007	2.0r7: Updated based on TNC-WG Nov 20 concall
12/21/2007	2.0r8: Updated based on TNC Review comments, Updated C header to reflect comments. Updated to fix minor issues discovered during creation of PB-TNC spec.
1/10/2008	2.0r9: Updated to fix issues discovered during TNC Review of PB-TNC spec.
1/23/2008	2.0r10: Updated to fix minor issues.

Open Issues

This section tracks issues for the duration of specification creation. This section will be removed prior to final draft.

#	Description	Comments
---	-------------	----------

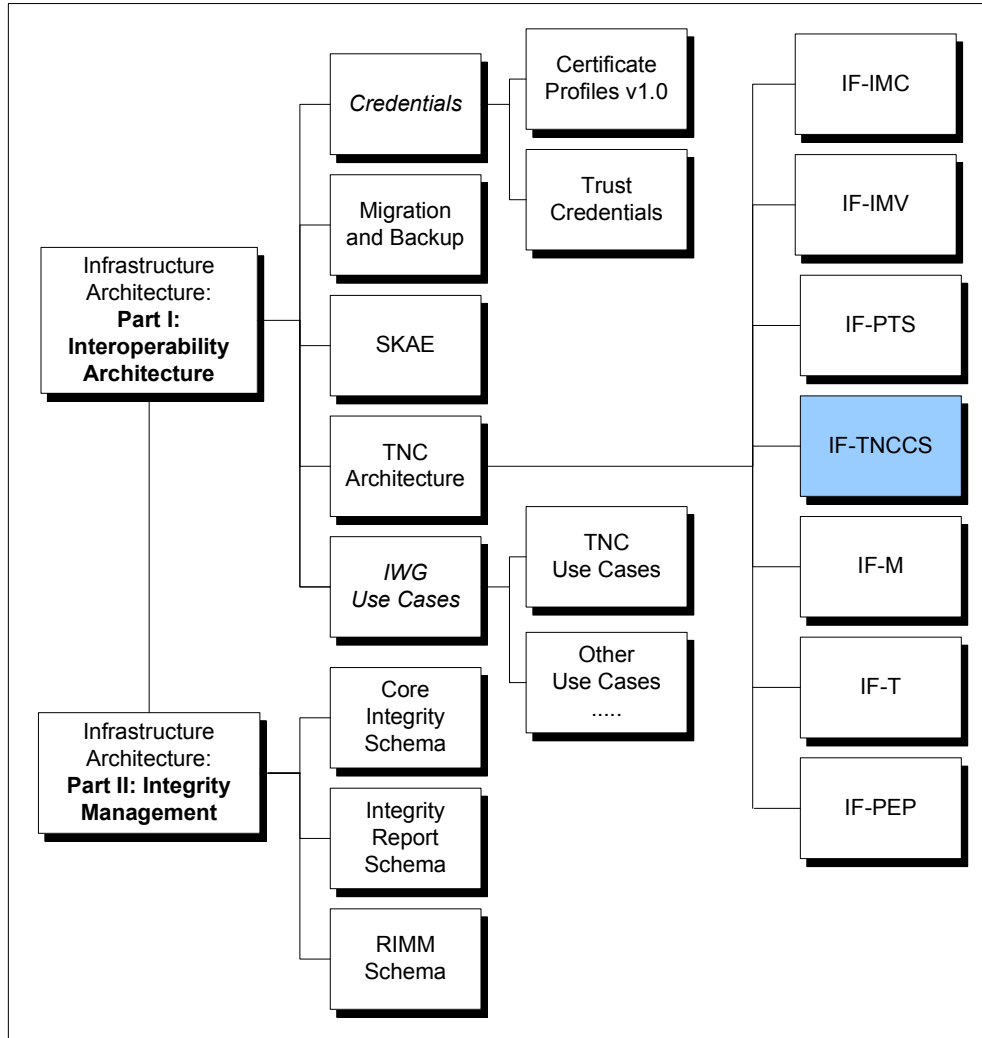
Closed Issues

This section will be removed prior to final draft.

#	Description	Comments
1	Should the TNCC expose a list of IMC (message types supported) to the other side? There is a privacy/vulnerability exposure concern. To make the exchange efficient TNCS could could ask the TNCC and the TNCC could use a bitmap to respond with (Yes No Do not want to Divulge)?	Discussed at TNC F2F 10/9/07. Agreed to leave this out for now.
2	Do we need a session id – a shorthand method to identify a session and its associated state across multiple interactions with a TNCCS?	Discussed at TNC F2F 10/9/07. Agreed can do with a TLV for session resumption later but need to make sure that session resumption requires proof of possession of a key securely derived from the earlier session. But we won't do this for now.
3	Should we add a feature that allows an IF-M message to be marked with an optional sender ID and recipient ID? This would allow an IMV to indicate that a particular message should only be delivered to a particular IMC, for instance. Right now, there is a feature like this in	Agreed on November 13, 2007 TNC-WG concall to do so.

	IF-M but it seems more natural to put it in IF-TNCCS since routing IF-M messages is part of the job of the TNCC and TNCS.	
4	Should we address the use case of a multi-user endpoint where the users prefer different languages?	We could easily address this use case if we allow the TNCC to send multiple TNCCS-Language-Preference messages in a single batch and state that this means that the TNCC has multiple language preferences (as would happen with multiple users). Why not? It's so easy and it will remove a non-supported use case that, while not common, is not that hard to imagine. Agreed on 11/20/07 TNC-WG call to address this use case.
5	Shall we move the example flows (section 3.5) to a separate Examples section at the end of the document?	Agreed on 11/20/07 TNC-WG call to do this. We'll put it after Security Considerations.
6	Should we retain the Sequence Number field?	Steve says: I don't think this is really necessary, since there's only ever one batch outstanding. Any error message must refer to the last batch sent by the party receiving the error message. I suggest that we change the Sequence Number field to be more reserved bits. Agreed on 11/20/07 TNC-WG call to do this.
7	Should we retain the ellipses ("...") after variable length fields?	Steve says: Does it really add anything? Agreed on 11/20/07 TNC-WG call that they'll be removed from IF-TNCCS 2.0 but we don't really need consistency with IF-M.
8	Why should the TNCC be required to implement the TNCCS-Access-Recommendation field?	Steve says: What's the harm if a TNCC doesn't care about the result so it just skips this message? Agreed on 11/20/07 TNC-WG call to make this optional.
9	Should we have a FATAL flag in the TNCCS-Error message?	Steve says: I think we should. That would allow us to signal non-fatal errors. Agreed on 11/20/07 TNC-WG call to add this. An example of a non-fatal error might be an ill-formed language tag.
10	The set of standard error codes needs some work.	Steve has written lots of comments on this section below. Agreed on 11/20/07 TNC-WG call to make these changes unless Ravi has any objections.
11	Appendix A (the C structures) needs to be updated.	Updated in IF-TNCCS 2.0r8.
12	IF-TNCCS Message Flows section need to be updated	Updated in IF-TNCCS 2.0r8.

TNC Document Roadmap



Acknowledgements

The TCG wishes to thank all those who contributed to this specification.

Aman Garg	3Com
Bipin Mistry	3Com
Scott Kelly	Aruba Networks
Amit Agarwal	Avaya
Mahalingam Mani	Avaya
Michael McDaniels	Extreme Networks
Hidenobu Ito	Fujitsu Limited
Houcheng Lee	Fujitsu Limited
Sung Lee	Fujitsu Limited
Kazuaki Nimura	Fujitsu Limited
Boris Balacheff	Hewlett-Packard
Mauricio Sanchez	Hewlett-Packard
Han Yin	Huawei
Diana Arroyo	IBM
Lee Terrell	IBM
Guha Prasad Venataraman	IBM
Chris Hessing	Identity Engines
Morteza Ansari	Infoblox
Stuart Bailey	Infoblox
Ivan Pulleyn	Infoblox
Ravi Sahita (Co-editor)	Intel Corporation
Ned Smith	Intel Corporation
Barbara Nelson	iPass
Chris Trytten	iPass
Roger Chickering	Juniper Networks
Charles Goldberg	Juniper Networks
Steve Hanna (Co-editor, TNC co-chair)	Juniper Networks
PJ Kirner	Juniper Networks
Lisa Lorenzin	Juniper Networks
Dean Sheffield	Juniper Networks
John Jerrim	Lancope
Gene Chang	Meetinghouse Data Communication

Alex Romanyuk	Meetinghouse Data Communication
John Vollbrecht	Meetinghouse Data Communication
Bernard Aboba	Microsoft Corporation
Mudit Goel	Microsoft Corporation
Ryan Hurst (Co-Editor)	Microsoft Corporation
Tom Kelnar	Microsoft Corporation
Paul Mayfield	Microsoft Corporation
Ram Vadali	Microsoft Corporation
Jun Wang	Microsoft Corporation
Sandilya Garimella	Motorola
Joseph Tardo	Nevis Networks
Pasi Eronen	Nokia Corporation
Meenakshi Kaushik	Nortel Networks
Ron Pon	Nortel Networks
Mike McCauley	Open Systems Consultants
Bryan Kingsford	Symantec Corporation
Paul Sangster (TNC co-chair)	Symantec Corporation
Curtis Simonson	University of New Hampshire InterOperability Labs
Brad Upson	University of New Hampshire InterOperability Labs
Lauren Giroux	U.S. National Security Agency
Chris Salter	U.S. National Security Agency
Jeff Six	U.S. National Security Agency
Rod Murchison	Vernier Networks
Michelle Sommerstad	Vernier Networks
Scott Cochrane	Wave Systems
Thomas Hardjono	Wave Systems
Greg Kazmierczak	Wave Systems

Table of Contents

1	Introduction	9
1.1	Scope and Audience	9
1.2	Keywords	9
2	Background	10
2.1	Role of IF-TNCCS	10
2.2	Supported Use Cases	10
2.3	Non-supported Use Cases	11
2.4	Requirements	11
2.5	Non-Requirements	12
2.6	Assumptions	13
2.7	Message Diagram Conventions	13
3	IF-TNCCS Protocol.....	14
3.1	Protocol Overview	14
3.2	IF-TNCCS 2.0 State Machine	14
3.3	Layering on IF-T	15
3.4	Example of IF-TNCCS Encapsulation.....	16
3.5	Interoperability with older IF-TNCCS versions	16
4	TLV Binding for IF-TNCCS 2.0.....	17
4.1	IF-TNCCS 2.0 Header.....	17
4.2	IF-TNCCS 2.0 Message.....	17
4.3	Standard IF-TNCCS 2.0 Message Types	19
4.4	TNCCS-Experimental.....	19
4.5	TNCCS-Batch-Type	20
4.6	TNCCS-IF-M-Message	21
4.7	TNCCS-Access-Recommendation	23
4.8	TNCCS-Remediation-Parameters	24
4.8.1	TCG Standard Remediation Parameter Types	25
4.9	TNCCS-Error.....	26
4.9.1	TCG Standard Error Codes.....	27
4.9.2	Error Parameters Structures for TCG Standard Error Codes	28
4.10	TNCCS-Language-Preference	28
4.11	TNCCS-Reason-String	29
5	Security Considerations	32
5.1	Threat Model	32
5.2	Countermeasures.....	32
6	Use Case Walkthrough.....	34
6.1	TNCC Initiated Assessment Use Case	34
6.1.1	IF-T Connection Setup	34
6.1.2	First TNCC Message.....	34
6.1.3	First TNCS Message	34
6.1.4	Second TNCC Message.....	34
6.1.5	TNCS Result.....	34
6.1.6	IF-T Connection Teardown.....	35
6.2	Sequence Diagram for TNCC Initiated Assessment.....	35
Appendix A: IF-TNCCS 2.0 C Structures.....		36
7	References	40
7.1	Normative References	40
7.2	Informative References.....	40

1 Introduction

1.1 Scope and Audience

The Trusted Network Connect Work Group (TNC-WG) has defined an open solution architecture that enables network operators to enforce policies regarding the security state of endpoints in order to determine whether to grant access to a requested network infrastructure. This security assessment of each endpoint is performed using a set of asserted integrity measurements covering aspects of the operational environment of the endpoint. Integrity measurements are carried between the TNC Client and TNC Server on a protocol called IF-TNCCS (Trusted Network Connect Client-Server), as shown in figure 1 below.

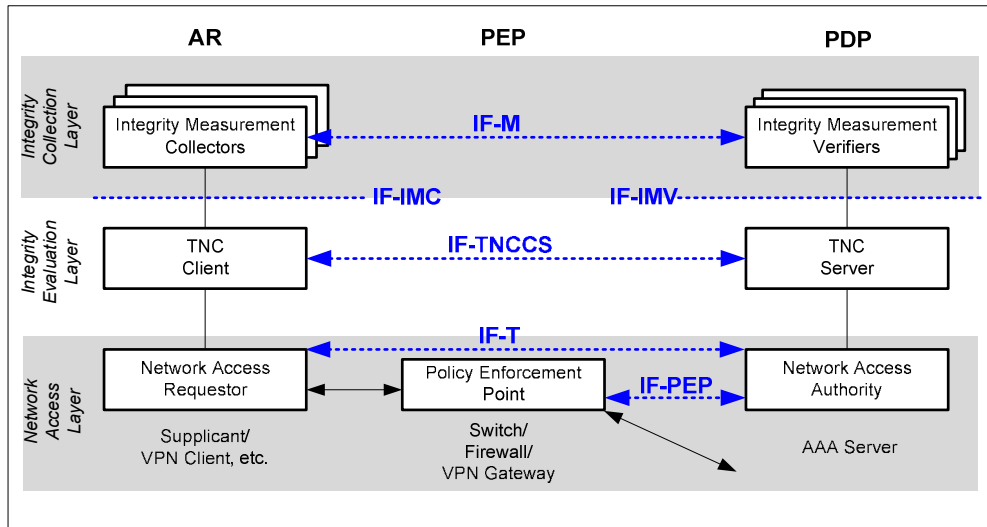


Figure 1 - TNC Architecture

This specification defines a new version of IF-TNCCS that is based on earlier versions of IF-TNCCS such as IF-TNCCS 1.1 [9] and IF-TNCCS-SOH 1.0 [10]. The goals of this new version of IF-TNCCS are:

- to support the same use cases and functionality as the earlier versions of IF-TNCCS
- to become the single agreed-upon standard client-server Network Access Control (NAC) protocol going forward

Before reading this document any further, the reader should review and understand the TNC Architecture specification [8]. If the reader is building a TNC Client that supports IF-IMC, the reader is encouraged to read the IF-IMC specification [5] prior to reading this document. If the reader is building a TNC Server that supports IF-IMV, the reader is encouraged to read the IF-IMV specification [6] prior to reading this document.

1.2 Keywords

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [1]. This specification does not distinguish blocks of informative comments and normative requirements. Therefore, for the sake of clarity, note that lower case instances of must, should, etc. do not indicate normative requirements.

2 Background

2.1 Role of IF-TNCCS

IF-TNCCS describes a standard way for the TNC Client and the TNC Server to exchange messages. More specifically, this interface defines a protocol and format for carrying:

- (a) Messages from Integrity Measurement Collectors (IMCs) to Integrity Measurement Verifiers (IMVs) (such as integrity measurements)
- (b) Messages from IMVs to IMCs (such as requests for additional integrity measurements, or remediation instructions)
- (c) Messages from TNC Clients to TNC Servers (such as control messages)
- (d) Messages from TNC Servers to TNC Clients (such as the IF-TNCCS Access Recommendation message)

Note that the contents of the messages being passed between the IMCs and IMVs ((a) and (b) above) are opaque to the IF-TNCCS layer. IF-TNCCS relies on the underlying transport protocol (IF-T) to provide a secure authenticated channel to protect the messages in transit between the TNC Client (TNCC) and the TNC Server (TNCS) and ensure they are delivered to the correct TNCC or TNCS.

2.2 Supported Use Cases

Use cases that IF-TNCCS supports are as follows.

TNCC Initiated Assessment Use Case

1. A TNCC initiates an integrity assessment (initial assessment or reassessment).
2. The IMCs send IF-M messages (typically integrity measurements). The TNCC receives these messages, collects them in a batch, and delivers them to the TNCS via the IF-TNCCS protocol. The TNCC may include standard and/or vendor-specific TNCC-TNCS messages in the same batch.
3. The TNCS receives the batch of messages. It processes any TNCC-TNCS messages itself and delivers the IF-M messages to the IMVs. After reviewing these messages, each IMV may provide an IMV Action Recommendation to the TNCS, indicating what action it recommends should be taken with respect to the endpoint's network access. Each IMV may also optionally respond by sending its own IF-M messages (remediation instructions, requests for more information, or other things) back to the IMCs. The TNCS delivers these messages (perhaps with some TNCC-TNCS messages) to the TNCC through IF-TNCCS.
4. This exchange of messages may continue for multiple round trips within a single integrity assessment. Eventually, the IF-M message exchange will end naturally or the TNCS will terminate it. The TNCS will send a final batch of messages to the TNCC, including the TNCS Action Recommendation.

TNCS Initiated Assessment Use Case

This is the same as the TNCC Initiated Assessment Use Case except that the first step is that the TNCS initiates an integrity assessment and the IMVs get to send messages first. Note that this differs from the current TNC architecture, where IMCs always send messages first. This change will avoid an extra round trip in scenarios where the IMVs need to start the exchange and the TNCS is initiating the handshake.

Language Preference

A TNCC informs the TNCS of the endpoint users' language preferences through IF-TNCCS using standard TNCC-TNCS message(s). This allows the TNCS and/or IMVs to adapt strings intended to be human-readable so they conform to the users' language preference before sending them to the TNCC.

Note that this version of IF-TNCCS includes support for multi-user endpoints where the users have different preferred languages. A TNCC can specify more than one preferred language. Strings can be sent from the TNCS to the TNCC in several languages.

Reason Strings

One or more IMVs provide reason strings to a TNCS, giving the reason for their IMV Action Recommendations. The TNCS passes these reason strings to the TNCC through IF-TNCCS using standard TNCC-TNCS message(s).

Multi-protocol TNC Client or TNC Server

In order to ensure a smooth transition from IF-TNCCS 1.X and IF-TNCCS-SOH 1.X to IF-TNCCS 2.0, a TNC Client or TNC Server wishes to support more than one of these protocols simultaneously.

2.3 Non-supported Use Cases

- None

2.4 Requirements

Here are the requirements that the IF-TNCCS 2.0 protocol must meet in order to successfully play its role in the TNC architecture. In addition to these requirements, requirements specified by the IETF NEA Working Group [11] must be met since compliance with those requirements is a goal of this specification.

- Meets the needs of the TNC architecture

The IF-TNCCS 2.0 protocol MUST support all the functions and use cases described in the TNC architecture as they apply to the relationship between the TNC Client and the TNC Server.

- Efficient

The TNC architecture delays network access until the endpoint is determined not to pose a security threat to the network based on its asserted integrity information. To minimize user frustration, the IF-TNCCS 2.0 protocol MUST minimize delays and make IMC-IMV communications as rapid and efficient as possible. Efficiency is also important when you consider that some network endpoints are small and low-powered and that some networks have high latency, high cost, or low bandwidth. Also, some transport protocols are half-duplex with a limited fragment size and require a full round trip per fragment.

- Extensible

IF-TNCCS will need to expand over time as new features are added to the TNC architecture. The IF-TNCCS 2.0 protocol MUST allow new features to be added easily, providing for a smooth transition and allowing newer and older architectural components to continue to work together. It MUST include support for vendor-specific extensions.

- Easy to use and implement

The IF-TNCCS 2.0 protocol MUST be easy for TNC Client and TNC Server vendors to use and implement. It should allow them to enhance existing products to support the TNC architecture and integrate legacy code without requiring substantial changes. The protocol should also make things easy for system administrators and end-users. Components of the TNC architecture should plug together automatically without requiring manual configuration.

- Transport Independent

The IF-TNCCS 2.0 protocol MUST be capable of operating over any IF-T [7] transport protocol, so long as this transport protocol meets the assumptions listed in section 2.6. Half-duplex and full-duplex transports MUST both be supported.

- Scalable

The IF-TNCCS 2.0 protocol MUST be highly scalable. For example, it MUST support having a large number (at least a hundred) IMCs and IMVs active in a single handshake and sending IF-M messages. The message type used for identifying and routing TNCC-TNCS and IF-M messages MUST support large numbers (hundreds) of standard defined message types and large numbers (hundreds) of vendor specific message types for each vendor with thousands of vendors. It MUST allow new message types to be defined over time.

- Able to coexist and function with IF-TNCCS 1.X and/or IF-TNCCS-SOH 1.X

In order to ensure a smooth transition from IF-TNCCS 1.X and IF-TNCCS-SOH 1.X to IF-TNCCS 2.0, the IF-TNCCS 2.0 protocol MUST be designed so that a TNC Client or TNC Server can support any and all of these protocols at once (e.g. by making it easy to quickly detect the difference between the different protocols). Note that this does not impose a requirement on any TNCC or TNCS to support multiple protocols. It simply enables a TNCC or TNCS to do so.

- Vendor neutral

The IF-TNCCS 2.0 protocol MUST be vendor neutral. However, it MUST include support for vendor-specific extensions.

- Fully interoperable

The IF-TNCCS 2.0 protocol MUST be designed so that any TNC Client and TNC Server that comply with the specification will interoperate (assuming that they support a common IF-T transport protocol and have a compatible set of IMCs and IMVs and policies).

- Internationalized

The IF-TNCCS 2.0 protocol MUST provide a way for the TNCC to inform the TNCS of the endpoint user's language preference using standard TNCC-TNCS message(s). Any strings intended to be human-readable MUST be adaptable so that they conform to the user's language preference.

- Reason String Support

The IF-TNCCS 2.0 protocol MUST provide a standard way for IMV and/or TNCS reason strings to be passed to the TNCC.

2.5 Non-Requirements

There are certain requirements that the IF-TNCCS 2.0 protocol explicitly is not required to meet. This list may not be exhaustive (complete).

- Fully compatible with existing IF-TNCCS 1.X or IF-TNCCS-SOH 1.X clients and servers

The IF-TNCCS 2.0 protocol does not need to be fully compatible with existing IF-TNCCS 1.X or IF-TNCCS-SOH 1.X clients and servers. That is, there is no expectation that a TNC Client that only supports IF-TNCCS 1.X or IF-TNCCS-SOH 1.X will work with a TNC Server that only supports IF-TNCCS 2.0. While it would be great if this was possible, this is probably not

consistent with the other requirements for IF-TNCCS 2.0. Therefore, it has been agreed that this is not a requirement. However, it is required (as noted above) that a TNC Client or TNC Server be able to support all of these protocols at once (or any ones that it may wish to support), in order to ensure a smooth transition from IF-TNCCS 1.X or IF-TNCCS-SOH 1.X to IF-TNCCS 2.0.

- Secure

IF-TNCCS 2.0 does not provide security services itself. Instead, it relies on IF-T for secure transport of messages between the TNC Client and the TNC Server. This includes authentication, encryption, integrity protection, and replay protection.

2.6 Assumptions

Here are the assumptions that the IF-TNCCS 2.0 protocol makes about other components in the TNC architecture.

- Format of integrity measurements

The format of the IMC-IMV messages that IF-TNCCS 2.0 conveys between the TNC Client and the TNC Server is opaque to IF-TNCCS 2.0. Each IMC-IMV message is simply represented as a binary piece of data within the IF-TNCCS 2.0 protocol.

- Transport

IF-T is the underlying transport protocol for ALL IF-TNCCS 2.0 communication. It is assumed that IF-T will provide a reliable transport mechanism, ensuring the timely delivery of IF-TNCCS 2.0 messages in the same order in which they were sent. It is also assumed that IF-T will secure all IF-TNCCS 2.0 communications, providing adequate authentication, encryption, integrity protection, and replay protection.

- Fragmentation

The IF-T protocol is expected to provide fragmentation when needed. However, it is understood that some IF-T protocols require a complete round trip for each fragment. Therefore, IF-TNCCS 2.0 will be designed to reduce fragmentation.

2.7 Message Diagram Conventions

This specification defines the syntax of the IF-TNCCS 2.0 messages using diagrams. Each diagram depicts the format and size of each field in bits. Implementations **MUST** send the bits in each diagram as they are shown, traversing the diagram from top to bottom and then from left to right within each line (which represents a 32-bit quantity). Multi-byte fields representing numeric values must be sent in network (big endian) byte order. Descriptions of bit fields (e.g. flags) values are described referring to the position of the bit within the field. These bit positions are numbered from the most significant bit through the least significant bit so a one byte field with only bit 0 set has the value 0x80.

3 IF-TNCCS Protocol

This section gives an overview of the IF-TNCCS 2.0 protocol.

3.1 Protocol Overview

The IF-TNCCS 2.0 protocol carries batches of IF-TNCCS messages between a TNC Client (TNCC) and a TNC Server (TNCS). It encapsulates IF-M messages and manages the TNCC-TNCS session. It runs over an IF-T transport protocol.

In order to work well over half-duplex IF-T protocols (such as those based on EAP [13]), IF-TNCCS 2.0 is a half-duplex protocol. The TNCC and TNCS take turns sending batches of messages to each other. While the half-duplex nature of IF-TNCCS 2.0 could slow exchanges that require many round trips or bidirectional multimedia exchanges, this is not a problem in practice because integrity checks do not typically involve multimedia or a large number of round trips. The benefit of working over half-duplex transports outweighs any limitations imposed.

Each IF-TNCCS 2.0 batch consists of a header followed by a sequence of IF-TNCCS 2.0 messages. Each IF-TNCCS 2.0 message has a Type-Length-Value (TLV) format with a few flags. The TLV format allows a recipient to skip messages that it does not understand.

This specification defines certain standard IF-TNCCS 2.0 message types. It also permits vendors to define their own vendor-specific message types. One of the most important standard IF-TNCCS 2.0 message types is TNCCS-IF-M. A message with this type contains an IF-M message. A TNCC or TNCS that receives such a message does not interpret the IF-M message within. Instead, it delivers the IF-M message to the appropriate set of Integrity Measurement Collectors (IMCs) or Integrity Measurement Verifiers (IMVs). Another important standard IF-TNCCS 2.0 message type is TNCCS-Batch-Type, which contains a batch type that drives state machine transitions.

A TNCS will often need to communicate with several TNCCs at once. The reverse may also be true, as when an endpoint has multiple network interfaces connected to different networks. Each TNCC-TNCS connection is instantiated as a separate IF-TNCCS session. There may be several sessions between a single TNCC/TNCS pair but this is unusual.

3.2 IF-TNCCS 2.0 State Machine

Figure 2 illustrates the state machine for IF-TNCCS 2.0, which shows the set of states that an IF-TNCCS 2.0 session can have and the possible transitions among these states. The following paragraphs describe this state machine in more detail.

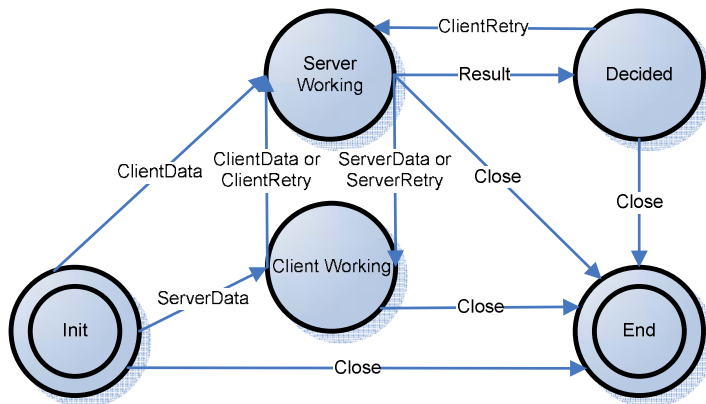


Figure 2: IF-TNCCS 2.0 state machine

Many state machine transitions are triggered by the transmission or reception of an IF-TNCCS 2.0 batch of a particular type. The type of an IF-TNCCS batch is indicated by the contents of an IF-TNCCS message of type TNCCS-Batch-Type within that batch. For brevity, this document says “a FOO batch” instead of “an IF-TNCCS 2.0 batch containing an IF-TNCCS message of type TNCCS-Batch-Type whose Batch Type is FOO”.

An IF-TNCCS 2.0 session starts in the Init state when the underlying transport protocol (IF-T) establishes a connection between a TNCC and a TNCS. If the TNCC initiated the underlying transport session, it starts by sending a ClientData batch to the TNCS, thus causing a transition to the Server Working state. If the TNCS initiated the transport session, it starts by sending an IF-TNCCS batch of type ServerData to the TNCC, thus causing a transition to the Client Working state.

The TNCC and TNCS may now alternate sending ClientData and ServerData batches to each other. Since IF-TNCCS 2.0 is a half-duplex protocol, only the TNCC can send a batch when the session is in the Client Working state and only the TNCS can send a batch when the session is in the Server Working state.

The most common way to end an exchange is for the TNCS to send a Result batch. This causes a transition into the Decided state. This is not a terminal state. The IF-T session remains open and another exchange can be initiated by having the TNCC send a ClientRetry batch. This can be useful when the TNCC (or more likely an IMC) discovers a suspicious condition on the endpoint, for example.

The TNCC can also initiate a new exchange by sending a ClientRetry batch when the session is in the Client Working state. The TNCS can perform a similar operation by sending a ServerRetry batch when the session is in the Server Working state. This can be useful if a suspicious condition arises on the endpoint or a policy changes on the PDP while an exchange is underway.

The only terminal state is the End state. This state is reached if the underlying IF-T connection closes. This can be caused by an action of the TNCC or TNCS or it can be caused by some external factor, such as pulling the network plug. No IF-TNCCS 2.0 batch is sent to indicate that the exchange has been closed. The TNCC and TNCS will generally receive some form of notification from the Network Access Requestor (NAR) and Network Access Authority (NAA) that the IF-T connection has been closed but this interaction is not standardized since the TNCC-NAR interface and the TNCS-NAA interface are not standardized. The Close notification causes the transition to the End state.

Note that a TNCC and TNCS may not always have exactly the same state for a given IF-TNCCS 2.0 session. For example, say that a session is in the Client Working state and the TNCC transmits a ClientData batch. While this batch is in transit (transmitted by the TNCC but not yet received by the TNCS), the TNCC will think that the session is in Server Working state but the TNCS will think that the session is in Client Working state. However, this is a temporary condition and does not cause problems in practice. The only possible issue is that a TNCC or TNCS does not know whether the other party has received its message until it receives a response from the other party.

Note that the TNCS cannot send a ServerRetry batch when the session is in the Decided state because the TNCS sent the most recent batch (the Result batch) and this would violate the half-duplex nature of the IF-TNCCS 2.0 protocol. Instead, a server that wishes to initiate a new exchange in the Decided state should close the IF-T connection and start a new IF-TNCCS 2.0 session.

3.3 Layering on IF-T

IF-TNCCS 2.0 batches are carried over protocol bindings of the IF-T protocol, which provides the interaction between a Network Access Requestor (NAR) and a Network Access Authority (NAA). IF-TNCCS 2.0 counts on IF-T to provide a secure transport. In particular, IF-T MUST support mutual authentication of the NAA and the NAR, confidentiality and integrity protection for IF-TNCCS 2.0 batches, and protection against replay attacks. IF-TNCCS is unaware of the underlying

transport protocols being used. IF-TNCCS operates directly on IF-T; no further layer of IF-TNCCS is expected.

3.4 Example of IF-TNCCS Encapsulation

This section shows how a typical IF-TNCCS 2.0 batch might be carried inside of IF-T For Tunneled EAP Methods.

Within the top-level IF-T header, the IF-TNCCS 2.0 header is packaged next, followed by a TNCCS-Batch-Type message and two TNCCS-IF-M messages that contain IF-M messages meant for the IMCs or IMVs on the platform.

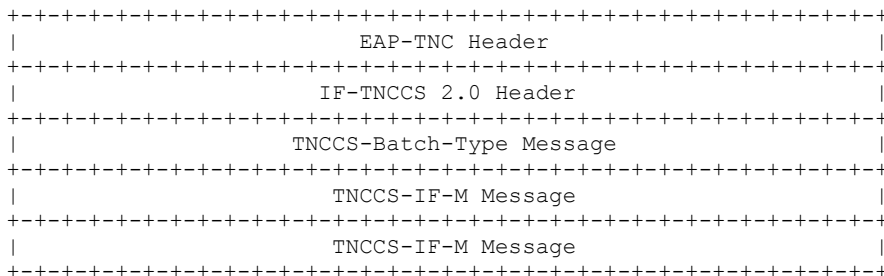


Figure 3: Example of EAP-TNC encapsulated IF-TNCCS message

3.5 Interoperability with older IF-TNCCS versions

A TNCC or TNCS may need to support IF-TNCCS 1.X, IF-TNCCS-SOH 1.X, and IF-TNCCS 2.0, all at the same time. To make this easier, the TLV Binding for IF-TNCCS 2.0 has been designed so that it can easily be distinguished from these other protocols. Simply looking at the first byte in the message allows one of these protocols to be distinguished from the others.

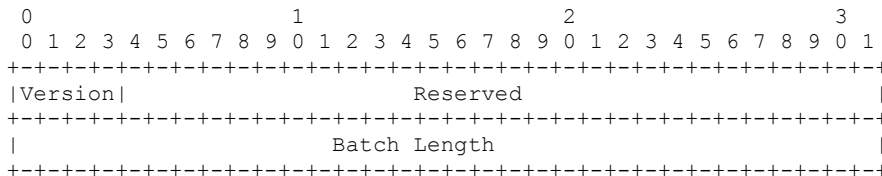
An IF-TNCCS 1.X message with UTF-8 encoding always begins with a byte with decimal value 60 (the '<' character that starts the XML document). An IF-TNCCS-SOH 1.X message always begins with a byte with decimal value 0 (the first byte of the SoH Header structure). An IF-TNCCS 2.0 batch encoded with the TLV Binding for IF-TNCCS 2.0 (this version of this specification) always begins with a byte with decimal value 32. Therefore, these three message formats can easily be distinguished. Future versions of IF-TNCCS should be carefully designed to ensure that messages can be easily distinguished by looking at the first byte.

4 TLV Binding for IF-TNCCS 2.0

This section contains the TLV Binding for IF-TNCCS 2.0. Other bindings for the IF-TNCCS 2.0 protocol can be described later, if necessary.

4.1 IF-TNCCS 2.0 Header

Every IF-TNCCS 2.0 batch MUST start with the following header. An IF-TNCCS 2.0 batch MUST contain only one instance of this header followed by one or more IF-TNCCS 2.0 messages. The IF-TNCCS 2.0 messages are defined in subsequent sections of this specification.



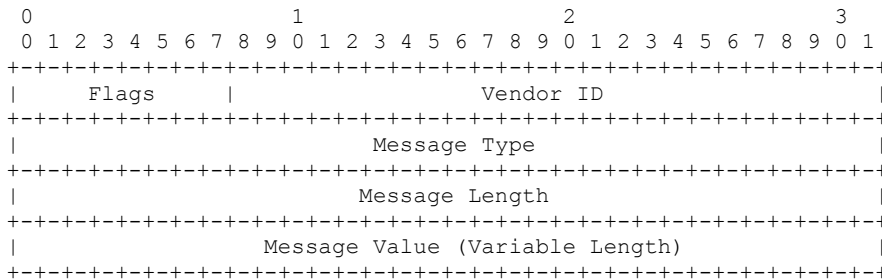
Version (4 bits): This field MUST be set to 2 when the batch conforms to this specification (IF-TNCCS 2.0). Later versions of IF-TNCCS may define other values for this field. If a TNCC or TNCS receives a Version value that it does not support, it SHOULD respond with an Invalid Parameter error code.

Reserved (28 bits): This field is reserved. For this version of this specification, it MUST be set to 0 on transmission and ignored on reception. Future versions of this specification may allow senders to set some of these bits and recipients to interpret them.

Batch Length (32 bits) – This length field contains the size of the full IF-TNCCS batch in octets. This length includes the IF-TNCCS 2.0 header and all the IF-TNCCS 2.0 messages in the batch. In other words, it includes the entire contents of the batch.

4.2 IF-TNCCS 2.0 Message

All IF-TNCCS 2.0 messages have the same overall structure, which is described in this section. Of course, the format and semantics of the Message Value field will vary, depending on the values of the Vendor ID and Message Type fields.



Flags (8 bits)

This field defines flags impacting the processing of this message.

Bit 0 of this flags field (the most significant bit) is known as the NOSKIP flag. If this flag is cleared (value 0), then the recipient (a TNCC or TNCS) may skip (ignore) this message if the message type is not understood or the recipient cannot or will not process the message as

required in the definition of that message. If this flag is set (value 1), then recipients MUST NOT skip this attribute.

This flag does not mean that all recipients must support this message. Instead, any recipient that receives a message with this flag set to 1 but cannot or will not process it as required MUST NOT act on any part of the IF-TNCCS batch. Instead, the recipient SHOULD include an Unsupported Mandatory Message error code in the next batch that it sends. In order to avoid taking action on some messages in a batch only to later find an unsupported NOSKIP flagged message, recipients of an IF-TNCCS 2.0 batch might choose to scan all of the messages in the batch prior to acting upon any of the messages, checking to determine whether one of them is an unsupported message with the NOSKIP flag set.

The other bits in this Flags field are reserved. For this version of IF-TNCCS 2.0, they MUST be set to 0 on transmission and ignored on reception.

Vendor ID (24 bits)

This Vendor ID field identifies a vendor by using the SMI Private Enterprise Number (PEN). Any organization can receive its own unique PEN from IANA, the Internet Assigned Numbers Authority. This Vendor ID qualifies the Message Type field so that each vendor has $2^{32}-1$ separate Message Types available for their use.

Message types standardized by the TCG must use the TCG SMI PEN (0x005597) in this field. Additionally, the Vendor ID 0xffffffff is reserved. A TNCC/TNCS MUST NOT send messages in which the Vendor ID has this reserved value (0xffffffff). If a TNCC or TNCS receives a message in which the Vendor ID field has this reserved value (0xffffffff), it SHOULD send an Invalid Parameter error code in the next batch that it sends.

TNC Clients and TNC Servers MUST NOT require support for particular vendor-specific extensions and MUST interoperate with other parties despite any differences in the set of vendor-specific extensions supported (although they MAY permit administrators to configure them to require support for specific extensions).

Message Type (32 bits)

This Message Type field identifies the type of the IF-TNCCS message contained in the Message Value field. The Message Type 0xffffffff is reserved. A TNCC/TNCS MUST NOT send messages in which the Message Type field has this reserved value (0xffffffff). If a TNCC or TNCS receives a message in which the Message Type field has this reserved value (0xffffffff), it SHOULD send an Invalid Parameter error code in the next batch that it sends. Unless otherwise prohibited in the definition of a particular IF-TNCCS 2.0 Message Type (e.g. TNCCS-Batch-Type), a single IF-TNCCS 2.0 batch may contain multiple messages with the same message type and/or Vendor ID.

The TCG and any other organization with a PEN can define $2^{32} - 1$ unique IF-TNCCS Message Types, as long as the organization's PEN is placed in the Vendor ID field of the message. Since the IF-TNCCS Message Type is qualified by the Vendor ID, there is no risk of conflicts as long as each organization uses its own PEN for the Vendor ID and manages its own set of $2^{32}-1$ message type values.

This document defines certain IF-TNCCS Message Types which, when used with the TCG SMI PEN, have standard meanings. Some of these IF-TNCCS Message Types are mandatory and therefore MUST be implemented by all TNCC and TNCS implementations that claim compliance with this specification. For details on which IF-TNCCS Message Types are mandatory, see the description of these message types later in section 4.

Note that the IF-TNCCS 2.0 Message Type field is completely separate from the IF-M Subtype field. The same value (e.g. 0) may have different meanings as an IF-TNCCS 2.0 message type and as an IF-M Subtype.

Message Length (32 bits)

This field specifies the length of this IF-TNCCS 2.0 message in octets. It includes this header (the fields Flags, Vendor ID, Message Type, and Message Length). Therefore, this value **MUST** always be at least 12.

Message Value (variable length)

The syntax and semantics of this field varies, depending on the values in the Vendor ID and Message Type fields. The syntax and semantics of several standard messages is defined in subsequent sections of this specification.

4.3 Standard IF-TNCCS 2.0 Message Types

This table provides a reference list with brief descriptions of the standard IF-TNCCS 2.0 message types to be used with the Vendor ID set to the TCG SMI PEN (0x005597). If these IF-TNCCS 2.0 message type values are used with a different Vendor ID, they have a completely different meaning that is not defined in this specification.

For more details on these message types, see the remainder of section 4. New message types may be added in future revisions of the protocol. For TCG standard IF-M Subtypes (which are completely different from IF-TNCCS 2.0 message types), please refer to the IF-M specification.

Message Type	Definition
0 (TNCCS20_TYPE_EXPERIMENT)	TNCCS-Experimental - reserved for experimental use
1 (TNCCS20_TYPE_BATCH)	TNCCS-Batch-Type - indicates the type of the IF-TNCCS 2.0 batch that contains this message
2 (TNCCS20_TYPE_IFM)	TNCCS-IF-M-Message - contains an IF-M message
3 (TNCCS20_TYPE_ACCESS_RECOMM)	TNCCS-Access-Recommendation - includes TNCS access recommendation
4 (TNCCS20_TYPE_REMED_PARAMS)	TNCCS-Remediation-Parameters - includes TNCS remediation parameters
5 (TNCCS20_TYPE_ERROR)	TNCCS-Error - error indicator
6 (TNCCS20_TYPE_LANG_PREF)	TNCCS-Language-Preference - sender's preferred language(s) for human-readable strings
7 (TNCCS20_TYPE_REASON_STR)	TNCCS-Reason-String - string explaining reason for TNCS access recommendation

4.4 TNCCS-Experimental

The TNCCS-Experimental IF-TNCCS 2.0 message type is an IF-TNCCS message type (value 0) that has been set aside for experimental purposes. It may be used to test code or for other experimental purposes. It **MUST NOT** be used in a production environment or in a product. This meaning for this IF-TNCCS message type only applies if the Vendor ID field in the IF-TNCCS Message Header contains the TCG SMI PEN (0x005597). If a different Vendor ID is contained in that field, the message type 0 has a completely different meaning not defined in this specification.

The Message Length and Message Value for this IF-TNCCS 2.0 message type are not specified. They may have almost any value, depending on what experiments are being conducted. Similarly, the Flags field for this message may have the NOSKIP bit set or cleared, depending on what

experiments are being conducted. However, note that the Message Length field must have a value of at least 12 since that is the total of the length of the fixed-length fields at the start of the IF-TNCCS message (the fields Flags, Vendor ID, Message Type, and Message Length).

A TNCC or TNCS implementation intended for production use **MUST NOT** send a message with this message type with the TCG SMI PEN as the Vendor ID. If it receives a message with this message type and with the TCG SMI PEN as the Vendor ID, it **SHOULD** ignore the message unless the NOSKIP bit is set, in which case it **SHOULD** send an Unsupported Mandatory Message error code in the next batch that it sends.

4.5 TNCCS-Batch-Type

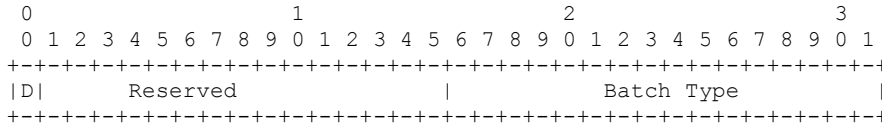
The IF-TNCCS 2.0 message type named TNCCS-Batch-Type (value 1) indicates the type of the IF-TNCCS 2.0 batch that contains it. This value is used to drive the state machine described in section 3.2.

Each IF-TNCCS batch **MUST** contain one and only one message with type TNCCS-Batch-Type. All TNCC and TNCS implementations **MUST** implement support for this IF-TNCCS message type.

The NOSKIP flag in the IF-TNCCS message header **MUST** be set for this message type and the Message Type field **MUST** contain 1. The Vendor ID field **MUST** contain the TCG SMI PEN (0x005597). If a different Vendor ID is contained in that field, the message type 1 has a completely different meaning not defined in this specification.

The Message Length field **MUST** contain the value 16 since that is the total of the length of the fixed-length fields at the start of the IF-TNCCS message (the fields Flags, Vendor ID, Message Type, and Message Length) along with the fields described below.

The following diagram illustrates the format and contents of the Message Value for this message type. The text after this diagram describes the fields in the Message Value.



Directionality (D) (1 bit)

When a TNCC is sending this message, the Directionality bit **MUST** be set to 0. When a TNCS is sending this message, the Directionality bit **MUST** be set to 1. This helps avoid any situation where two TNCCs or two TNCSs engage in a dialog. It also helps with debugging.

Reserved (15 bits)

These Reserved bits **MUST** be set to 0 on transmission and ignored on reception.

Batch Type (16 bits)

The Batch Type field **MUST** have one of the values from the following table. If any other value is received, the recipient **MUST** send an Invalid Parameter error code in response. In addition, if the value received is not permitted for the current state, according to the state machine in section 3.2, the recipient **MUST** send an Unexpected Batch Type error code in response.

Batch Type	Batch Type Name	Definition
------------	-----------------	------------

Number		
1	TNCCS20_BT_CLIENT_DATA	The TNCC may send a batch with this Batch Type to convey messages to the TNCS. A TNCS MUST NOT send this Batch Type. This message may be the only message in a batch, if the TNCC has nothing else to send.
2	TNCCS20_BT_SERVER_DATA	The TNCS may send a batch with this Batch Type to convey messages to the TNCC. A TNCC MUST NOT send this Batch Type. This message may be the only message in a batch, if the TNCS has nothing else to send.
3	TNCCS20_BT_RESULT	The TNCS may send a batch with this Batch Type to indicate that it has completed its evaluation. The batch SHOULD include a TNCCS-Access-Recommendation message.
4	TNCCS20_BT_CLIENT_RETRY	The TNCC may send a batch with this Batch Type to indicate that it wishes to restart an exchange. A TNCS MUST NOT send this Batch Type. This message may be the only message in a batch, if the TNCC has nothing else to send.
5	TNCCS20_BT_SERVER_RETRY	The TNCS may send a batch with this Batch Type to indicate that it wishes to restart the exchange. A TNCC MUST NOT send this Batch Type. This message may be the only message in a batch, if the TNCS has nothing else to send.

4.6 TNCCS-IF-M-Message

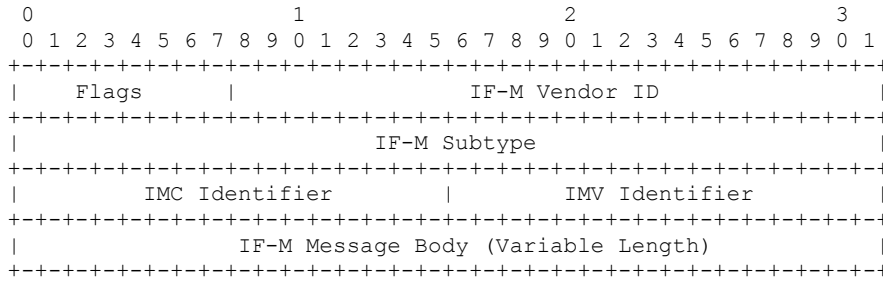
The IF-TNCCS 2.0 message type named TNCCS-IF-M-Message (value 2) contains one IF-M message. Many batches will contain several TNCCS-IF-M-Message messages but some batches may not contain any messages of this type.

All TNCC and TNCS implementations MUST implement support for this IF-TNCCS message type. Generally, this support will consist of forwarding the enclosed IF-M message to the appropriate IMCs and IMVs. Specific requirements are contained later in the description of this message type.

The NOSKIP flag in the IF-TNCCS message header MUST be set for this message type. The Vendor ID field MUST contain the TCG SMI PEN (0x005597) and the Message Type field MUST contain 2. If a non-zero value is contained in the Vendor ID field, message type 2 has a completely different meaning not defined in this specification.

The Message Length field MUST contain the length of the entire IF-TNCCS message, including the fixed-length fields at the start of the IF-TNCCS message (the fields Flags, Vendor ID, Message Type, and Message Length), the fixed-length fields listed below (Flags, IF-M Vendor ID, IF-M Message Subtype, IMC Identifier, and IMV Identifier), and the IF-M Message Body. Since the IF-M Message Body is variable length, the value in the Message Length field will vary also. However, it MUST always be at least 24 to cover the fixed-length fields listed in the preceding sentences.

The following diagram illustrates the format and contents of the Message Value for this message type. The text after this diagram describes the fields in the Message Value.



Flags (8 bits)

This field defines flags relating to the IF-M message.

Bit 0 of this flags field (the most significant bit) is known as the EXCL flag (for exclusive). If the EXCL bit is cleared (value 0), the TNCC or TNCS that receives this IF-TNCCS message SHOULD deliver the IF-M message contained in this IF-TNCCS message to all IMCs or IMVs that have expressed an interest in IF-M messages with this IF-M Vendor ID and IF-M Subtype. If a TNCC receives a message with the EXCL flag set (value 1), the TNCC SHOULD deliver the IF-M message contained in this IF-TNCCS message only to the IMC identified by the IMC Identifier field. However, if the identified IMC has not expressed an interest in IF-M messages with this IF-M Vendor ID and IF-M Subtype, the IF-M message should be silently discarded. Analogous requirements apply to a TNCS that receives a message with the EXCL flag set.

The EXCL bit allows, for example, an IMV to handle the circumstance where there are two IMCs on the endpoint that are interested in a particular kind of IF-M messages and the IMV has remediation instructions that only apply to one of those IMCs.

The other bits in this Flags field are reserved. For this version of IF-TNCCS 2.0, they MUST be set to 0 on transmission and ignored on reception.

IF-M Vendor ID (24 bits)

The IF-M Vendor ID field identifies a vendor by using the SMI Private Enterprise Number (PEN). Any organization can receive its own unique PEN from IANA, the Internet Assigned Numbers Authority. The IF-M Vendor ID qualifies the IF-M Subtype field so that each vendor has 2^32-1 separate IF-M Subtypes available for their use. The IF-M Vendor ID 0xffffffff is reserved. A TNCC or TNCS MUST NOT send messages in which the IF-M Vendor ID has this reserved value (0xffffffff). If a TNCC or TNCS receives a message in which the IF-M Vendor ID field has this reserved value (0xffffffff), it SHOULD send an Invalid Parameter error code in the next batch that it sends.

IF-M messages standardized by the TCG always use the TCG SMI PEN (0x005597) in this field. For detailed descriptions of those messages, see the IF-M specification [12].

IF-M Subtype (32 bits)

The IF-M Subtype field identifies the type of the IF-M message contained in the IF-M Message Body field. The IF-M subtype 0xffffffff is reserved. A TNCC or TNCS MUST NOT send messages in which the IF-M subtype field has this reserved value (0xffffffff). If a TNCC or TNCS receives a message in which the IF-M Subtype field has this reserved value (0xffffffff), it SHOULD send an Invalid Parameter error code in the next batch that it

sends. A TNCC or TNCS MUST support having multiple IF-M messages that have the same IF-M Subtype and/or IF-M Vendor ID contained in a single IF-TNCCS 2.0 batch.

Note that the IF-TNCCS 2.0 Message Type field is completely separate from the IF-M Subtype field. The same value (e.g. 0) may have different meanings as an IF-TNCCS 2.0 message type and as an IF-M Subtype.

IMC Identifier (16 bits)

The IMC Identifier field contains the identifier of the IMC associated with this IF-M message.

The TNCC MUST assign a unique IMC Identifier value (but not `0xffff`) to each IMC involved in a message exchange and include this IMC identifier in this field for any IF-M messages sent by that IMC. The IMC Identifier value assigned to an IMC by a TNCC MUST NOT change during the course of a message exchange. TNCCs that use the IF-IMC API SHOULD use the least significant 16 bits from the `imcID` provided by the TNCC during the `TNC_IMC_Initialize()` call (described in the IF-IMC specification [5]).

When a TNCS sets the EXCL flag for an IF-M message, the TNCS MUST set the IMC Identifier field to the identifier of the IMC that should receive the IF-M message. If the EXCL flag is not set, a TNCS MAY still set the IMC Identifier value for IF-M messages that it sends to indicate that the IF-M message is intended as a response to a message sent by the IMC associated with the specified IMC Identifier. If the TNCS does not wish to indicate any IMC in this manner, it SHOULD set this field to the reserved value `0xffff`.

IMV Identifier (16 bits)

The IMV Identifier field contains the identifier of the IMV associated with this IF-M message.

The TNCS MUST assign a unique IMV Identifier value (but not `0xffff`) to each IMV involved in a message exchange and include this IMV identifier in this field for any IF-M messages sent by that IMV. The IMV Identifier value assigned to an IMV by a TNCS MUST NOT change during the course of a message exchange. TNCSs that use the IF-IMV API SHOULD use the least significant 16 bits from the `imvID` provided by the TNCS during the `TNC_IMV_Initialize()` call (described in the IF-IMV specification [6]).

When a TNCC sets the EXCL flag for an IF-M message, the TNCC MUST set the IMV Identifier field to the identifier of the IMV that should receive the IF-M message. If the EXCL flag is not set, a TNCC MAY still set the IMV Identifier value for IF-M messages that it sends to indicate that the IF-M message is intended as a response to a message sent by the IMV associated with the specified IMV Identifier. If the TNCS does not wish to indicate any IMV in this manner, it SHOULD set this field to the reserved value `0xffff`.

IF-M Message Body (variable length)

The IF-M Message Body field contains the body of the IF-M message that is being carried in this IF-TNCCS message. The length of this field can be determined by subtracting the length of the fixed-length fields at the start of the IF-TNCCS message (the fields Flags, Vendor ID, Message Type, and Message Length) and the fixed-length fields at the start of the TNCCS-IF-M-Message (Flags, IF-M Vendor ID, IF-M Message Subtype, IMC Identifier, and IMV Identifier) from the Message Length contained in the IF-TNCCS message's Message Length field. The length of these fixed-length fields is 24 octets. Therefore, any TNCC or TNCS that receives an TNCCS-IF-M-Message with an IF-TNCCS Message Length field whose value is less than 24 SHOULD send an Invalid Parameter error code in response.

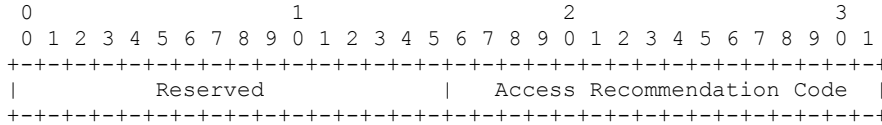
4.7 TNCCS-Access-Recommendation

The IF-TNCCS 2.0 message type named TNCCS-Access-Recommendation (value 3) is used by the TNCS to provide an access recommendation after the TNCS has completed some assessment of the Access Requestor (AR). The TNCS SHOULD include one message of this type in any batch of type Result and SHOULD NOT include a message of this type in any other type of batch. The

TNCC MUST NOT send an IF-TNCCS message with this message type. The TNCC SHOULD implement and process this message and SHOULD ignore any message with this message type that is not part of a batch of type Result.

The NOSKIP flag in the IF-TNCCS message header MUST NOT be set for this message type. The Vendor ID field MUST contain the TCG SMI PEN (0x005597) and the IF-TNCCS Message Type field MUST contain 3. If a non-zero value is contained in the Vendor ID field, message type 3 has a completely different meaning not defined in this specification. The Message Length field MUST contain the value 16 since that is the total of the length of the fixed-length fields at the start of the IF-TNCCS message (the fields Flags, Vendor ID, Message Type, and Message Length) along with the Access Recommendation field described below.

The following diagram illustrates the format and contents of the Message Value for this message type. The text after this diagram describes the fields in the Message Value.



Reserved (16 bits)

These Reserved bits MUST be set to 0 on transmission and ignored on reception.

Access Recommendation Code (16 bits):

The Access Recommendation Code field identifies the Access Recommendation that the TNCS has made for this TNCC at this time. This field MUST have one of these three values: 1 for Access Allowed (full access), 2 for Access Denied (no access), or 3 for Quarantined (partial access). If a TNCC receives an Access Recommendation Code value other than these three values, it SHOULD respond with an Invalid Parameter error code. Other values may be defined in future versions of IF-TNCCS but only if the IF-TNCCS version number is changed.

4.8 TNCCS-Remediation-Parameters

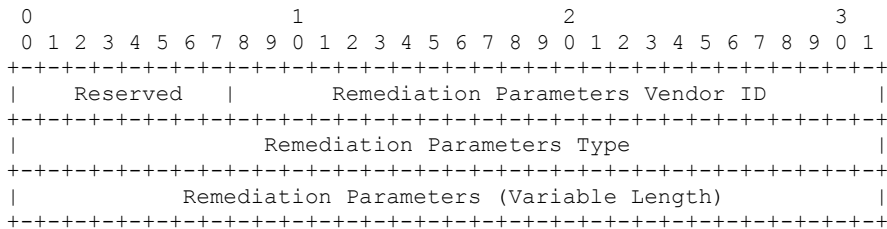
The IF-TNCCS 2.0 message type named TNCCS-Remediation-Parameters (value 4) is used by the TNCS to provide global (not IMV-specific) remediation parameters after the TNCS has completed some assessment of the Access Requestor (AR). The TNCS MAY include one or more messages of this type in any batch of any type but this message type is most useful in batches of type Result.

The TNCC MUST NOT send an IF-TNCCS message with this message type. The TNCC may implement and process this message but is not required to do so. It may skip this message. Even if the TNCC implements this message type, it is not obligated to act on it.

The NOSKIP flag in the IF-TNCCS message header MUST NOT be set for this message type. The Vendor ID field MUST contain the TCG SMI PEN (0x005597) and the IF-TNCCS Message Type field MUST contain 4. If a non-zero value is contained in the Vendor ID field, message type 4 has a completely different meaning not defined in this specification.

The Message Length field MUST contain the length of the entire IF-TNCCS message, including the fixed-length fields at the start of the IF-TNCCS message (the fields Flags, Vendor ID, Message Type, and Message Length), the fixed-length fields listed below (Reserved, Remediation Parameters Vendor ID, and Remediation Parameters Type), and the Remediation Parameters. Since the Remediation Parameters field is variable length, the value in the Message Length field will vary also. However, it MUST always be at least 20 to cover the fixed-length fields listed in the preceding sentences.

The following diagram illustrates the format and contents of the Message Value for this message type. The text after this diagram describes the fields in the Message Value.



Reserved (8 bits)

These Reserved bits MUST be set to 0 on transmission and ignored on reception.

Remediation Parameters Vendor ID (24 bits)

The Remediation Parameters Vendor ID field identifies a vendor by using the SMI Private Enterprise Number (PEN). Any organization can receive its own unique PEN from IANA, the Internet Assigned Numbers Authority. The Remediation Parameters Vendor ID qualifies the Remediation Parameters Type field so that each vendor has 2^32 separate Remediation Parameters Types available for their use.

Remediation Parameters Types standardized by the TCG always use the TCG SMI PEN (0x005597) in this field. A list of TCG-standardized Remediation Parameters Types appears later in this section.

Remediation Parameters Type (32 bits)

The Remediation Parameters Type field identifies the type of remediation parameters contained in the Remediation Parameters field. A TNCC or TNCS MUST support having multiple Remediation Parameters messages contained in a single IF-TNCCS 2.0 batch that have the same Remediation Parameters Type and/or Remediation Parameters Vendor ID.

Note that the Remediation Parameters Type is completely separate from the IF-TNCCS 2.0 message type and the IF-M Subtype field. The same value (e.g. 0) may have different meanings in each of these fields.

Remediation Parameters (variable length)

The Remediation Parameters field contains the actual remediation parameters carried in this IF-TNCCS message. The length of this field can be determined by subtracting the length of the fixed-length fields at the start of the IF-TNCCS message (the fields Flags, Vendor ID, Message Type, and Message Length) and the fixed-length fields at the start of the TNCCS-Remediation-Parameters message (Reserved, Remediation Parameters Vendor ID, and Remediation Parameters Type) from the Message Length contained in the IF-TNCCS message's Message Length field. The length of these fixed-length fields is 20 octets. Therefore, any TNCC that receives a TNCCS-Remediation-Parameters message with an IF-TNCCS Message Length field whose value is less than 20 SHOULD consider this a malformed message. The TNCC may send an Invalid Parameter error code in response, if this is practical according to the IF-TNCCS 2.0 state machine. In many cases, it will not be practical since the TNCCS-Remediation-Parameters message often comes in a batch of type Result and according to the IF-TNCCS 2.0 state machine, a TNCC cannot send a batch after this except a ClientRetry batch, which would restart the handshake. That is generally not desirable.

4.8.1 TCG Standard Remediation Parameter Types

This subsection defines several Remediation Parameter Types that have been standardized by TCG.

TNC-Remediation-URI

This Remediation Parameter Type is employed by creating a TNCCS-Remediation-Parameters message with a Remediation Parameters Vendor ID equal to the TCG SMI PEN (0x005597) and a Remediation Parameters Type of 1. The Remediation Parameters field in the TNCCS-Remediation-Parameters message MUST contain a URI, as described in RFC 3986 [4]. This URI contains instructions and resources for remediation. The TNCC MAY load the URI and display the resulting web page to the user. The TNCC may also ignore the URI or take another action with it. The TNCS and any other parties involved in configuring this remediation URI should consider the likely capabilities of the TNCC when creating the URI and the content referenced by the URI. For example, they should consider the TNCC's language preferences as expressed in the TNCCS-Language-Preference message.

TNC-Remediation-String

This Remediation Parameter Type is employed by creating a TNCCS-Remediation-Parameters message with a Remediation Parameters Vendor ID equal to the TCG SMI PEN (0x005597) and a Remediation Parameters Type of 2. The Remediation Parameters field in the TNCCS-Remediation-Parameters message MUST contain a UTF-8-encoded string. This string should contain human-readable instructions for remediation. The TNCC MAY display the instructions to the user. The TNCC may also ignore the instructions or take another action with them. The TNCS and any other parties involved in configuring these instructions should consider the likely capabilities of the TNCC when creating the instructions. For example, they should consider the TNCC's language preferences as expressed in the TNCCS-Language-Preference message.

Other TCG standard Remediation Parameter Types may be defined in later revisions of this specification.

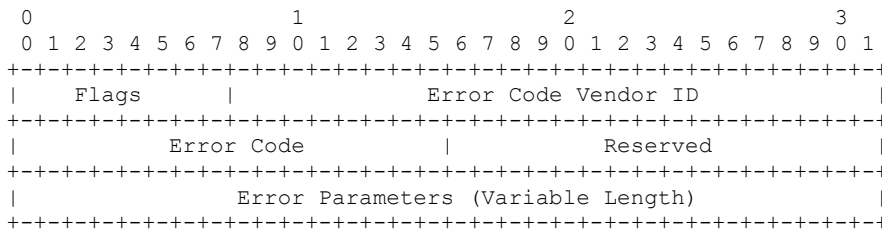
4.9 TNCCS-Error

The IF-TNCCS 2.0 message type named TNCCS-Error (value 5) is used by the TNCC or TNCS to indicate that an error has occurred. The TNCC or TNCS MAY include one or more messages of this type in any batch of any type. Other messages may also be included in the same batch. The party that receives a TNCCS-Error message MAY log it or take other action as deemed appropriate. If the FATAL flag is set (value 1), the recipient MUST close the IF-TNCCS session after processing the batch without sending any messages in response. Every TNCC and TNCS MUST implement this message type.

The NOSKIP flag in the IF-TNCCS message header MUST be set for this message type. The Vendor ID field MUST contain the TCG SMI PEN (0x005597) and the IF-TNCCS Message Type field MUST contain 5. If a non-zero value is contained in the Vendor ID field, message type 5 has a completely different meaning not defined in this specification.

The Message Length field MUST contain the length of the entire IF-TNCCS message, including the fixed-length fields at the start of the IF-TNCCS message (the fields Flags, Vendor ID, Message Type, and Message Length), the fixed-length fields listed below (Flags, Error Code Vendor ID, Error Code, and Reserved), and the Error Parameters. Since the Error Parameters field is variable length, the value in the Message Length field will vary also. However, it MUST always be at least 20 to cover the fixed-length fields listed in the preceding sentences.

The following diagram illustrates the format and contents of the Message Value for this message type. The text after this diagram describes the fields in the Message Value.



Flags (8 bits)

This field defines flags relating to the error.

Bit 0 of this flags field (the most significant bit) is known as the FATAL flag. If the FATAL bit is cleared (value 0), the TNCC or TNCS that receives this IF-TNCCS message SHOULD process this error and then continue with the exchange. If the FATAL flag is set (value 1), the TNCC or TNCS that receives this IF-TNCCS message MUST terminate the exchange after processing the error.

The FATAL bit allows a TNCC or TNCS to signal a fatal error (like an invalid batch type) and/or a non-fatal error (like an invalid language tag for a preferred language).

The other bits in this Flags field are reserved. For this version of IF-TNCCS 2.0, they MUST be set to 0 on transmission and ignored on reception.

Error Code Vendor ID (24 bits)

The Error Code Vendor ID field identifies a vendor by using the SMI Private Enterprise Number (PEN). Any organization can receive its own unique PEN from IANA, the Internet Assigned Numbers Authority. The Error Code Vendor ID qualifies the Error Code field so that each vendor has 2^{16} separate Error Codes available for their use.

Error codes standardized by the TCG always use the TCG SMI PEN (0x005597) in this field. For detailed descriptions of those messages, see the next few subsections.

Error Code (16 bits)

The Error Code field identifies the type of error being signaled with this message. The format of the Error Parameters field depends on the value of the Error Code Vendor ID and the Error Code. However, any recipient that does not understand a particular error code can process the error fairly well by using the FATAL flag to determine whether the error is fatal and the IF-TNCCS Message Length to skip over the Error Parameters field (or log it).

Reserved (16 bits)

The Reserved bits MUST be set to 0 on transmission and ignored on reception.

4.9.1 TCG Standard Error Codes

The following error codes are TCG standard error codes, hence the Error Code Vendor ID MUST be the TCG SMI PEN (0x005597). The following table defines the 16 bit Error Code. Vendor-specific error codes may be defined by setting the Error Code Vendor ID to the defining vendor's SMI PEN and setting the Error Code field to whatever error code(s) that vendor has defined. The format, length, and meaning of the Error Parameters field varies, based on the Error Code Vendor ID and Error Code. Subsequent sections of this document define the format, length, and meaning of the Error Parameters for the TCG standard error codes defined in this section.

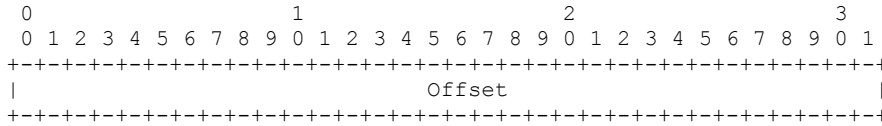
Error Code	Definition
0 (TNCCS20_ERR_UNEXP_BATCH)	Unexpected Batch Type. Error Parameters has offset of offending TNCCS-Batch-Type message.
1 (TNCCS20_ERR_INVALID_PARAM)	Invalid Parameter. Error Parameters has offset where invalid value was found.
2 (TNCCS20_ERR_LOCAL_ERROR)	Local Error. Error Parameters are empty.

3 (TNCCS20_ERR_UNSUP_MAND_MSG)	Unsupported Mandatory Message. Error Parameters has offset of offending IF-TNCCS Message
--------------------------------	--

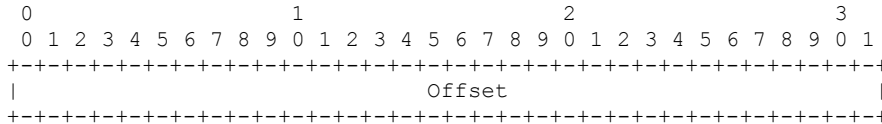
4.9.2 Error Parameters Structures for TCG Standard Error Codes

This section defines the format, length, and meaning of the Error Parameters field for the TCG standard error codes defined in this specification.

The Error Parameters field has the following structure for the TCG standard error code 0. The Offset field is the offset in octets from the start of the IF-TNCCS batch to the TNCCS-Batch-Type message whose type was not recognized. The FATAL flag MUST be set for this error code.

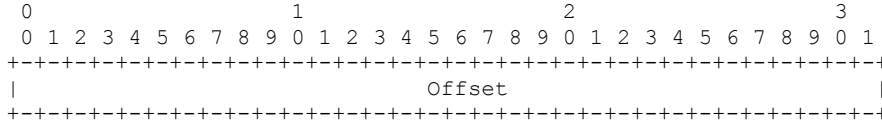


The Error Parameters field has the following structure for the TCG standard error code 1. The Offset field is the offset in octets from the start of the IF-TNCCS batch to the invalid value. The FATAL flag may either be set or cleared for this error code.



The Error Parameters field is zero length for the TCG standard error code 2. The FATAL flag MUST be set for this error code.

The Error Parameters field has the following structure for the TCG standard error code 3. The Offset field is the offset in octets from the start of the IF-TNCCS batch to the IF-TNCCS message whose message type was not recognized (and where the NOSKIP flag was set) The FATAL flag MUST be set for this error code.



4.10 TNCCS-Language-Preference

The IF-TNCCS 2.0 message type named TNCCS-Language-Parameters (value 6) is used by the TNCC to indicate which language or languages it would prefer for any human-readable strings that might be sent to it. This allows the TNCS and IMVs to adapt any messages they may send to the TNCC's preferences (probably determined by the language preferences of the user(s) of the Access Requestor).

The TNCS may also send this message type to the TNCC to indicate the TNCS's language preferences but this is not very useful since the TNCC rarely sends human-readable strings to the TNCS and, if it does, rarely can adapt those strings to the preferences of the TNCS.

No TNCC or TNCS is required to send or implement this message type. However, a TNCS SHOULD attempt to adapt to user language preferences by implementing this message type, passing the language preference information to IMVs, and allowing administrators to configure human-readable languages in whatever languages are preferred by their users.

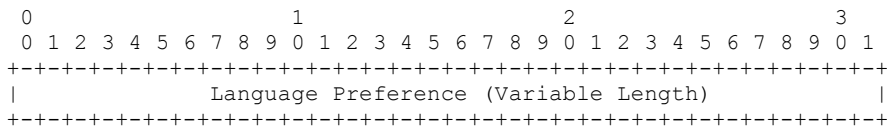
A TNCC or TNCS may include a message of this type in any batch of any type. However, it is suggested that this message be included in the first batch sent by the TNCC or TNCS in an IF-TNCCS 2.0 session so that the recipient can start adapting its human-readable messages as soon as possible. If one TNCCS-Language-Parameters message is received and then another one is received in a later batch for the same IF-TNCCS 2.0 session, the value included in the later message should be considered to replace the value in the earlier message.

A TNCC or TNCS **MUST NOT** include more than one message of this type in a single batch. If a TNCC or TNCS receives more than one message of this type in a single batch, it should ignore all but one of them.

The NOSKIP flag in the IF-TNCCS message header **MUST NOT** be set for this message type. The Vendor ID field **MUST** contain the TCG SMI PEN (0x005597) and the IF-TNCCS Message Type field **MUST** contain 6. If a non-zero value is contained in the Vendor ID field, message type 6 has a completely different meaning not defined in this specification.

The Message Length field **MUST** contain the length of the entire IF-TNCCS message, including the fixed-length fields at the start of the IF-TNCCS message (the fields Flags, Vendor ID, Message Type, and Message Length) and the Language Preference field. Since the Language Preference field is variable length, the value in the Message Length field will vary also. However, it **MUST** always be at least 12 to cover the fixed-length fields listed in the preceding sentences.

The following diagram illustrates the format and contents of the Message Value for this message type. The text after this diagram describes the fields in the Message Value.



Language Preference (variable length)

The Language Preference field contains an Accept-Language header, as described in RFC 3282 [2] (US-ASCII only, no control characters allowed, no NUL termination). Any TNCC or TNCS that sends a TNCCS-Language-Preference message **MUST** ensure that the Language Preference field conforms to this format.

A zero length Language Preference field indicates that no language preference information is available. Generally, there's no need to send a TNCCS-Language-Preference message with a zero length Language Preference field since this is equivalent to sending no TNCCS-Language-Preference message at all but it may be useful to send a zero length Language Preference field if a TNCCS-Language-Preference message with a non-zero length Language Preference field was sent in an earlier batch but these preferences no longer apply.

4.11 TNCCS-Reason-String

The IF-TNCCS 2.0 message type named TNCCS-Reason-String (value 7) is used by the TNCS to provide a human-readable explanation for an access recommendation conveyed in a TNCCS-Access-Recommendation message. Therefore, a TNCCS-Reason-String message **SHOULD** only be included in the same batch as a TNCCS-Access-Recommendation message. The TNCC **MUST NOT** ever send a TNCCS-Reason-String message.

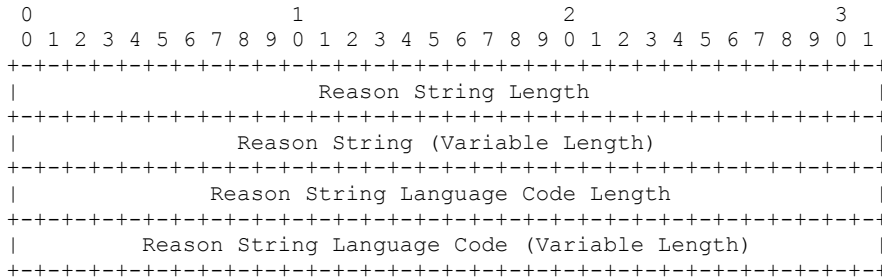
The TNCC is not required to implement this message type and the TNCS is not required to send it. However, there is some benefit to doing so since users are often curious about why network access was denied (if that was the access recommendation). Since the strings contained in this message are human-readable, the TNCS **SHOULD** adapt them to the TNCC's language preferences as expressed in any TNCCS-Language-Preference message sent by the TNCC in this IF-TNCCS 2.0 session.

A TNCS MAY include more than one message of this type in any batch of any type. However, it is suggested that this message be included in the same batch as a TNCCS-Access-Recommendation message. If more than one TNCCS-Reason-String message is included in a single batch, the TNCC SHOULD consider the strings included in these messages to be equivalent in meaning. This allows the TNCS to return multiple equivalent reason strings in different languages, which may help if the TNCS is not able to accommodate the TNCC's language preferences.

The NOSKIP flag in the IF-TNCCS Message Header MUST NOT be set for this message type. The Vendor ID field MUST contain the TCG SMI PEN (0x005597) and the IF-TNCCS Message Type field MUST contain 7. If a non-zero value is contained in the Vendor ID field, message type 7 has a completely different meaning not defined in this specification.

The Message Length field MUST contain the length of the entire IF-TNCCS message, including the fixed-length fields at the start of the IF-TNCCS message (the fields Flags, Vendor ID, Message Type, and Message Length), the fixed-length fields listed below (Reason String Length and Reason String Language Code Length), and the Reason String and Reason String Language Code fields. Since the Reason String and Reason String Language Code fields are variable length, the value in the Message Length field will vary also. However, it MUST always be at least 20 to cover the fixed-length fields listed in the preceding sentences. In fact, the Message Length field MUST be exactly the sum of 20 (for the fixed-length fields) and the values of the Reason String Length and Reason String Language Code Length fields. If this is not the case, the recipient SHOULD send an Invalid Parameter error code in response, if this is practical according to the IF-TNCCS 2.0 state machine. In many cases, it will not be practical since the TNCCS-Reason-String message often comes in a batch of type Result and according to the IF-TNCCS 2.0 state machine, a TNCC cannot send a batch after this except a ClientRetry batch, which would restart the handshake. That is generally not desirable.

The following diagram illustrates the format and contents of the Message Value for this message type. The text after this diagram describes the fields in the Message Value.



Reason String Length (32 bits)

The Reason String Length field contains the length of the Reason String field in octets.

Reason String (variable length)

The Reason String field contains a UTF-8 encoded string that provides a human-readable reason for the TNCS Access Recommendation. NUL termination MUST NOT be included. A zero length string SHOULD NOT be sent since this is the same as sending no reason string at all, leaving the reason unspecified.

Reason String Language Code Length (32 bits)

The Reason String Language Code Length field contains the length of the Reason String Language Code field in octets.

Reason String Language Code (variable length)

The Reason String Language Code field contains a US-ASCII string containing a well-formed RFC 4646 [2] language tag that indicates the language(s) used in the Reason String in this message. NUL termination **MUST NOT** be included in this field. A zero length string **MAY** be sent for this field (essentially omitting this field) to indicate that the language code for the reason string is not known.

5 Security Considerations

As described in the Assumptions section of this document, IF-T is assumed to provide reliable and secure transport for the IF-TNCCS 2.0 protocol (including authentication, confidentiality, integrity protection, and replay protection). Still, it is useful to describe the possible threats to IF-TNCCS and the countermeasures that are employed. This section does that.

5.1 Threat Model

There are several possible threats to the IF-TNCCS 2.0 protocol.

Untrusted intermediaries on the network between the Access Requestor and the Policy Decision Point may attempt to observe data sent between the TNC Client and TNC Server via IF-TNCCS 2.0, modify this data in transit, reorder it, or replay it. They may also attempt to mount a denial of service attack against either party or truncate the exchange prematurely. If successful, these attacks may result in improper access decisions relating to the Access Requestor, failure to reassess access control decisions in light of changed circumstances, improper remediation instructions sent to the Access Requestor (which could lead to the compromise of the Access Requestor), unauthorized access to confidential information about the Access Requestor's health and/or identity, improper reason strings or other messages that might be displayed to the user, access to reusable credentials such as posture assertions, denial of service on the Access Requestor, and even complete denial of access to the network (if a denial of service attack against the Policy Decision Point is successful).

Trusted intermediaries between the Access Requestor and the Policy Decision Point include the Network Access Requestor and the Network Access Authority. These parties are considered trusted because they are responsible for properly implementing the security protections provided by IF-T. If they fail to do so properly, these security protections may be diminished or eliminated altogether. The possible attacks are the same as those listed in the previous paragraph. To give one fairly likely example, if a Network Access Requestor fails to properly authenticate and authorize the Network Access Authority (whether through implementation error or through user configuration to "trust anyone"), the improperly authorized Network Access Authority may mount any of the previously described attacks against the Access Requestor.

Compromise of any of the trusted parties (the TNC Client, the Network Access Requestor, the Network Access Authority, or the Policy Decision Point) may result in failures that are equivalent to those listed in the first paragraph. These failures may be even more dangerous since they will not be detectable by observing network traffic or by examining and comparing audit logs. Failure to properly secure communications between the TNC Client and the Network Access Requestor or between the TNC Server and the Network Access Authority is usually indistinguishable from compromise of those parties. Compromise of the operating system or other critical software, firmware, or hardware components on the Access Requestor or Policy Decision Point will typically result in an equivalent result. And an attacker's ability to gain privileged access to the Access Requestor or Policy Decision Point (even for a brief time, long enough to disable or misconfigure security settings) is generally equivalent as well. If the Access Requestor or Policy Decision Point are dependent on other services for their proper operation (including Integrity Measurement Collectors, Integrity Measurement Verifiers, directories, and patch management services), compromise of those services may result in compromise or failure of the dependent parties. Of course, compromise or failure of Policy Decision Point components is most serious since this would probably affect a large number of Access Requestors while the effects of Access Requestor compromise might well be limited to a single machine.

5.2 Countermeasures

The primary countermeasure against attacks by untrusted network intermediaries is the security provided by the IF-T protocol. Any candidate IF-T protocols should be carefully examined to ensure that all the threats described above are adequately addressed.

As noted above, compromise or erroneous operation of any of the trusted parties is a serious matter with substantial security implications. This includes the TNC Client, the TNC Server, the Network Access Requestor, and the Network Access Authority. These are all security-sensitive components so they should be built and managed in accordance with best practices for security devices. This is especially important for the Policy Decision Point since a compromise of this device would affect the security and availability of the entire network (similar to compromise of a AAA server). Communications between the trusted parties must also be secured. For example, if the TNC Server and the Network Access Authority are separate components, their communications must be secured.

Since the Access Requestor may be a mobile device with little physical security (such as a laptop computer or even a public telephone), it should generally be assumed that some proportion of Access Requestors will be compromised and therefore hostile. The Policy Decision Point should be designed to be robust against hostile Access Requestors. Once a compromised Access Requestor is detected, it can be treated in a manner equivalent to an untrusted party and should pose no greater threat than any other untrusted party.

Countermeasures against a compromised Policy Decision Point (or a component thereof such as a TNC Server or a Network Access Authority) include prevention of compromise, detection of compromise, and mitigation of the effects of compromise. For prevention, the Policy Decision Point should be implemented using secure implementation techniques (e.g. secure coding and minimization) and managed using secure practices (e.g. strong authentication and separation of duty). For detection, the behavior of the Policy Decision Point should be monitored (e.g. via logging especially of remediation instructions, intrusion detection systems, and probes that impersonate a valid Access Requestor and record Policy Decision Point behavior) and any anomalies analyzed. For mitigation, Access Requestors should not blindly follow remediation instructions received from a trusted Policy Decision Point. At least for patches and other dangerous actions, they should validate these actions (e.g. via user confirmation) before proceeding. It should not be possible to configure an Access Requestor to trust all Policy Decision Points without proper authentication and authorization.

6 Use Case Walkthrough

This section provides informative (non-binding) walkthroughs of one typical IF-TNCCS use case, showing how IF-TNCCS 2.0 supports this use case. The walkthrough describes one particular example IF-TNCCS 2.0 exchange. It does not contain any normative text but is intended to serve solely as an example. A sequence diagram that illustrates this walkthrough is included at the end of this section.

6.1 TNCC Initiated Assessment Use Case

In this use case, the TNCC initiates an assessment. After exchanging a few IF-TNCCS batches, the TNCS decides to recommend full access.

6.1.1 IF-T Connection Setup

The first step is to have the NAR establish the IF-T connection. Of course, this is out of scope for IF-TNCCS but it is a necessary prerequisite since the IF-T connection must be established in order to carry IF-TNCCS traffic. IF-T connection establishment should include mutual authentication of the AR and PDP and establishment of a secure channel between them that will be used to carry the IF-TNCCS handshake. The TNCC and TNCS will both be in the Init state at the end of this step.

6.1.2 First TNCC Message

Since this is a TNCC initiated assessment, the TNCC sends the first IF-TNCCS batch. This batch contains an IF-TNCCS 2.0 header followed by three IF-TNCCS 2.0 messages. The first message is a TNCCS-Batch-Type with a Batch Type of ClientData. To be more precise, it is an IF-TNCCS 2.0 message with a message type of TNCCS-Batch-Type and a Message Value whose Batch Type field has the value ClientData. The second message is a TNCCS-IF-M-Message containing an IF-M message revealing the endpoint's OS type (which the TNCC would have obtained from the relevant IMC). The third message is a TNCCS-Language-Preference that specifies the user's language preferences (e.g. "Accept-Language: en-US"). After the TNCS receives this batch, both the TNCC and TNCS will be in the Server Working state.

6.1.3 First TNCS Message

The TNCS responds by sending a batch that requests further information about the endpoint. The first message in the batch is a TNCCS-Batch-Type with a Batch Type of ServerData. The second message in the batch is a TNCCS-IF-M-Message containing an IF-M message inquiring about the posture of the anti-virus component (presumably sent by an anti-virus IMV). After the TNCC receives this batch, both the TNCC and TNCS will be in the Client Working state.

6.1.4 Second TNCC Message

The TNCC responds by sending a batch containing the requested information about the posture of the anti-virus component. The first message in the batch is a TNCCS-Batch-Type with a Batch Type of ClientData. The second message in the batch is a TNCCS-IF-M-Message containing an IF-M message containing the requested information about the posture of the anti-virus component (presumably sent by the IMC responsible for the anti-virus component). After the TNCS receives this batch, both the TNCC and TNCS will be in the Server Working state.

6.1.5 TNCS Result

The TNCS responds by sending a batch containing the TNCCS-Access-Recommendation determined by the TNCS. The first message in the batch is a TNCCS-Batch-Type with a Batch Type of Result. The second message in the batch is a TNCCS-Access-Recommendation that indicates the TNCS' access recommendation. In this example, the TNCS has decided to recommend full network access so the Access Recommendation Code field will have the value 1 for Access Allowed. After the TNCC receives this batch, both the TNCC and TNCS will be in the Decided state.

6.1.6 IF-T Connection Teardown

Once the IF-TNCCS session is complete, the IF-T connection can be closed if it is now longer needed. This will result in a Close notification to the TNCC and TNCS. After that notification is received, both the TNCC and TNCS will be in the End state.

6.2 Sequence Diagram for TNCC Initiated Assessment

The following sequence diagram (Figure 4) illustrates the TNCC Initiated Assessment use case, as described in section 6.1.

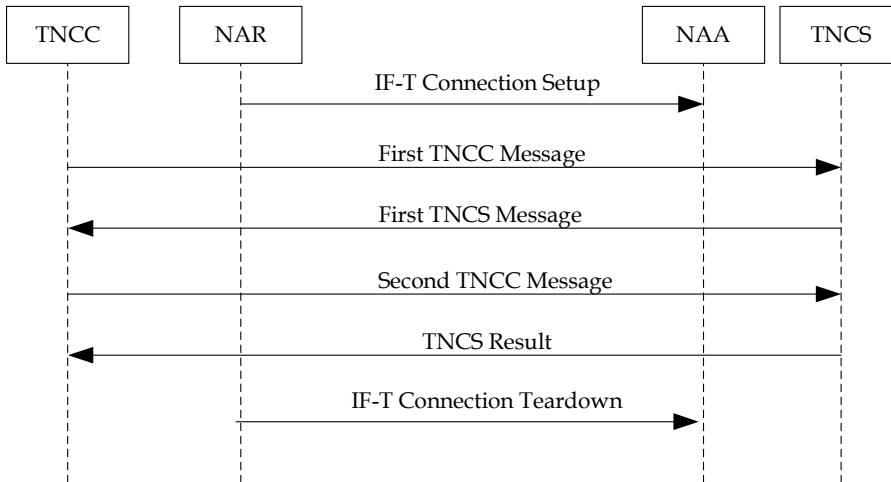


Figure 4 – TNCC Initiated Assessment Sequence Diagram

Appendix A: IF-TNCCS 2.0 C Structures

This section provides a C header file for the TLV binding of the IF-TNCCS 2.0 protocol.

```
/* tnc_if_tnccs_2_0.h
 *
 * Trusted Network Connect IF-TNCCS protocol version 2.0
 * December 27, 2007
 *
 * Copyright(c) 2007, Trusted Computing Group, Inc. All rights
 * reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * • Redistributions of source code must retain the above copyright
 * notice, this list of conditions and the following disclaimer.
 *
 * • Redistributions in binary form must reproduce the above copyright
 * notice, this list of conditions and the following disclaimer in
 * the documentation and/or other materials provided with the
 * distribution.
 *
 * • Neither the name of the Trusted Computing Group nor the names of
 * its contributors may be used to endorse or promote products
 * derived from this software without specific prior written
 * permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
 * COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
 * BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
 * ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 *
 * Contact the Trusted Computing Group at
 * admin@trustedcomputinggroup.org for information on specification
 * licensing through membership agreements.
 *
 * Any marks and brands contained herein are the property of their
 * respective owners.
 */

#ifndef _TNCCS_2_0_
#define _TNCCS_2_0_

#define IANA_SMI_PEN_TCG 0x005597

/*****
 * TNCCS 2.0 TLV Header
 *****/
```

```

*****/

typedef struct _tag_tnccs20_header {
    unsigned int u4_version:4;
    unsigned int u28_reserved:28;
    unsigned int u32_batch_length;
} t_s_tnccs20_header;

/*****
    TNCCS 2.0 Message
*****/

typedef enum _tag_tnccs20_type {
    TNCCS20_TYPE_EXPERIMENT = 0,
    TNCCS20_TYPE_BATCH,
    TNCCS20_TYPE_IFM,
    TNCCS20_TYPE_ACCESS_RECOMM,
    TNCCS20_TYPE_REMED_PARAMS,
    TNCCS20_TYPE_ERROR,
    TNCCS20_TYPE_LANG_PREF,
    TNCCS20_TYPE_REASON_STR
} t_e_tnccs20_type;

typedef struct _tag_tlv_msg {
    unsigned int u8_flags:8;
    unsigned int u24_vendor_id:24; /* use IANA_SMI_PEN_TCG for TNC*/
    unsigned int u32_msg_type;
    unsigned int u32_msg_length;
    unsigned char p_u8_msg_value[1]; /* actually variable size */
} t_s_tnccs20_tlv_msg;

/*****
    TNCCS Batch-type Message
*****/

typedef enum _tag_tnccs20_batch_type {
    TNCCS20_BT_CLIENT_DATA = 1,
    TNCCS20_BT_SERVER_DATA,
    TNCCS20_BT_RESULT,
    TNCCS20_BT_CLIENT_RETRY,
    TNCCS20_BT_SERVER_RETRY
} t_e_tnccs20_batch_type;

typedef enum _tag_tnccs20_ctrl_dirn {
    TNCCS20_CTRL_TNCC_TO_TNCS =0,
    TNCCS20_CTRL_TNCS_TO_TNCC =1
} t_e_tnccs20_ctrl_dirn;

typedef struct _tag_tnccs20_tlv_batch {
    unsigned int u1_batch_dirn:1; /*use t_e_tnccs20_ctrl_dirn*/
    unsigned int u15_reserved:15;
    unsigned int u16_batch_type:16; /*use t_e_tnccs20_batch_type*/
} t_s_tnccs20_tlv_batch;

/*****

```

```
Standard IF-M Message
*****/

#define TNCCS20_IFM_FLAG_EXCL 0x80

typedef struct _tag_tnccs20_tlv_ifm {
    unsigned int u8_ifm_flags:8;
    unsigned int u24_ifm_vendor_id:24; /* use IANA_SMI_PEN_TCG for
TNC*/
    unsigned int u32_ifm_subtype;
    unsigned int u16_ifm_imc_id:16;
    unsigned int u16_ifm_imv_id:16;
    unsigned char p_u8_ifm_value[1]; /* actually variable size */
} t_s_tnccs20_tlv_ifm;

/*****
Access Recommendation message
*****/

typedef enum _tag_tnccs20_access_type {
    TNCCS20_ACCESS_ALLOWED = 1,          /*FULL ACCESS*/
    TNCCS20_ACCESS_DENIED,              /*NO ACCESS*/
    TNCCS20_ACCESS_QUARANTINE           /*PARTIAL ACCESS*/
} t_e_tnccs20_access_type;

typedef struct _tag_tnccs20_tlv_access {
    unsigned int u16_reserved:16;
    unsigned int u16_access_type; /*t_e_tnccs20_access_type*/
} t_s_tnccs20_tlv_access;

/*****
Remediation parameters
*****/

typedef enum _tag_tnccs20_remed_type {
    TNCCS20_REMED_PARAM_URI = 1,
    TNCCS20_REMED_STRING
} t_e_tnccs20_remed_type;

typedef struct _tag_tnccs20_tlv_remed {
    unsigned int u8_remed_flags:8;
    unsigned int u24_remed_vendor_id:24; /*IANA_SMI_PEN_TCG for TNC*/
    unsigned int u32_remed_param_type; /*use t_e_tnccs20_remed_type*/
    unsigned char p_u8_remed_params[1]; /* actually variable size */
} t_s_tnccs20_tlv_remed;

/*****
Error message TLV and sub-structures
*****/

typedef enum _tag_tnccs20_error_code {
    TNCCS20_ERR_UNEXP_BATCH = 0,
    TNCCS20_ERR_INVALID_PARAM,
    TNCCS20_ERR_LOCAL_ERROR,
```

Formatted: German (Germany)

```
|      TNCCS20_ERR_UNSUP_MAND_MSG
} t_e_tnccs20_error_code;

#define TNCCS20_ERR_FLAG_FATAL 0x80

typedef struct _tag_tnccs20_tlv_error {
    unsigned int u8_err_flags:8;
    unsigned int u24_err_vendor_id:24; /*IANA_SMI_PEN_TCG for TNC*/
    unsigned int u16_err_code:16; /*use t_e_tnccs20_error_code*/
    unsigned int u16_reserved:16;
    unsigned char p_u8_err_params[1]; /* actually variable size */
} t_s_tnccs20_tlv_error;

typedef unsigned int t_u32_tnccs20_err_offset_param;

/*****
Language Preference
*****/

typedef unsigned char t_s_tnccs20_tlv_lang[1]; /*as per RFC 3282*/

/*****
Reason strings
*****/

typedef struct _tag_tnccs20_tlv_reason {
    unsigned int u32_reason_strlen;
    unsigned char p_u8_reason_str[1]; /* actually variable size */
    unsigned int u32_reason_langcode_len;
    unsigned char p_u8_reason_langcode[1]; /* actually variable size */
} t_s_tnccs20_tlv_reason;

#endif // _TNCCS_2_0_
```

7 References

7.1 Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>
- [2] Phillips, A. and M. Davis, "Tags for the Identification of Languages", Internet Engineering Task Force RFC 4646, September 2006.
- [3] Alvestrand, H., "Content Language Headers", Internet Engineering Task Force RFC 3282, May 2002.
- [4] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", Internet Engineering Task Force RFC 3986, January 2005.

7.2 Informative References

- [5] Trusted Computing Group, *TNC IF-IMC*, Specification Version 1.2, February 2007.
- [6] Trusted Computing Group, *TNC IF-IMV*, Specification Version 1.2, February 2007.
- [7] Trusted Computing Group, *TNC IF-T for Tunneled EAP Methods*, Specification Version 1.1, May 2007.
- [8] Trusted Computing Group, *TNC Architecture for Interoperability*, Specification Version 1.2, May 2007.
- [9] Trusted Computing Group, *TNC IF-TNCCS*, Specification Version 1.1, February 2007
- [10] Trusted Computing Group, *TNC IF-TNCCS-SOH*, Specification Version 1.0, May 2007
- [11] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", Work In Progress, November 2007.
- [12] Trusted Computing Group, *TNC IF-M*, Specification Version 1.0, Work In Progress.
- [13] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowitz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.