ROSETTANET
eBusiness Standards for the Global Supply Chain

Technical Recommendation

RosettaNet Implementation Framework V02.00.01

High Availability Features

Issue 01.00.00
17 December 2004

Table of Contents

1 Document Management

1.1 Legal Disclaimer

RosettaNet™, its members, officers, directors, employees, or agents shall not be liable for any injury, loss, damages, financial or otherwise, arising from, related to, or caused by the use of this document or the specifications herein, as well as associated guidelines and schemas. The use of said specifications shall constitute your express consent to the foregoing exculpation.

1.2 Copyright

1.3 Trademarks

RosettaNet, Partner Interface Process, PIP and the RosettaNet logo are trademarks or registered trademarks of “RosettaNet,” a non-profit organization. All other product names and company logos mentioned herein are the trademarks of their respective owners. In the best effort, all terms mentioned in this document that are known to be trademarks or registered trademarks have been appropriately recognized in the first occurrence of the term.

1.4 Acknowledgements

RosettaNet acknowledges the following companies for contributing towards this document:

- Cisco Systems
- Fujitsu
- Hewlett Packard
- Intel
- NTT Communications
- Sony EMCS
- Sterling Commerce

1.5 Related Documents

- RosettaNet Implementation Framework: Core Specification 2.0 [RNIF20]

1.6 Purpose

RosettaNet Implementation Framework 2.0 (RNIF 2.0) describes some failure handling procedures. This Technical Recommendation (TR) describes additional procedures for failure handling.

1.7 Scope

This document contains information describing enhancements to the RNIF 2.0 Specification regarding failure handling. This document does not contain any other RNIF changes or information regarding PIPs.

1.8 Conformance Statement

Compliance to the enhancements described in this TR is mandatory, only if failure handling procedures for the situations described in this specification is available in an RNIF 2.0 implementation. Applications that conform to this TR MUST still conform to all requirements of [RNIF20] and relevant Technical Advisories unless explicitly overruled in this TR.

1.9 Document Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

The terms PIP Instance, PIP Initiator, and PIP Responder are defined in [RNIF20].

1.10 Document Version History

Version	Date	Description
Issue 1.00.00	17 December 2004	Publication to the membership

2 Introduction

This Technical Recommendation (TR) prescribes changes to the RosettaNet Implementation Framework 2.0 (RNIF 2.0) for some failure handling situations.

2.1 Terms

The terms Action, Signal, PIP Instance, PIP Initiator, PIP Responder are defined in [RNIF20].

2.2 Issue

This Technical Recommendation enhances the techniques described in Section 2.6.4 of RNIF 2.0 Specification to address different types of communication failures. The immediate concern is the ability to address temporary failures caused by a large number of PIP transactions.

3 High Availability Features

This Technical Recommendation (TR) describes a limited number of ways a failure can occur while communicating RNIF messages with trading partners; and recommends procedures to recover from such failures. Any of the following failures could happen during a RNIF based communication.

1. Transfer Protocol failure (e.g., HTTP)
2. (RNIF) message failure (e.g., non-receipt of an expected Receipt Acknowledgement)
3. (RNIF) implementation failure (e.g., a server hosting the RNIF implementation fails)

This TR addresses some of these failures. The failures could be permanent or temporary. While it may be possible to recover from some failures based on the techniques described in this paper, recovery is not guaranteed in all situations.

3.1 Business Case

RNIF messaging service is currently installed in hundreds of sites. The software solution that implements RNIF uses a finite amount of hardware resources, and its ability to handle peak load of connections (sometimes in excess of 50K messages per hour) can vary based on the specific hardware or the vendor of the solutions.

Under these circumstances, RNIF will be left responding with “busy” signals. Sometimes, low-end RNIF solutions may simply drop connections, leaving the peer RNIF at the other side of the Internet wondering what happened. It has been observed by some RNIF implementers that by defining standard behavior for such failures could help trading partners recover from failures more quickly and improve overall messaging performance with moderate hardware resources. Note: These issues are not specific to RosettaNet, and can happen on any HTTP traffic.

There should be a solution that addresses peak load issues and brings closure dropped connections. RosettaNet would like to address this problem and provide a solution in a standardized way to promote interoperability among all RNIF implementations.

3.2 Attributes of Failures

Failures may be anticipated or unanticipated. Some failures may be caused by scheduled service shutdowns. If the communicating partner is unaware of such shutdowns, the communicating partner may assume a failure. Similarly, a change of system configuration at a site may cause disruption of communication with that site, making the communicating partner to assume a failure. This TR does not distinguish between these anticipated and unanticipated failures.

Failure is Relative: A "failure" is relative to a Trading Partner (TP), in that TP A may perceive a failure at TP B, while Sender TP C may not perceive a failure at TP B within the same time period!

Before or After: Failures may exist prior to creating a PIP Instance, or a failure may happen during the execution of a PIP Instance. Thus, either PIP Initiator or PIP Responder may fail during a PIP message exchange.

One or Both: PIP Initiator and PIP Responder may also fail simultaneously. Therefore, the solution described must address single or multiple failures of the PIP Initiator and PIP Responder. In this TR, we do not distinguish between PIP Initiator and PIP Responder. Instead we use the roles (message) Sender and (message) Receiver, because either PIP Initiator or PIP Responder could be a sender or a receiver for different messages.

3.2.1 Transfer Protocol Failure

The only Transfer Protocol we consider in this TR is HTTP.

In RNIF2.0, two HTTP response codes, 200 or 202 (Section 2.4.2.2 "Processing Inbound HTTP Posts") MUST be returned for a successful HTTP communication. Response code 200 is used for synchronous HTTP requests and 202 is used for asynchronous HTTP requests.

RNIF 2.0 states that "3xx, 4xx and 5xx error conditions must be dealt with in the usual way, governed by the local policy." This TR describes a standardized manner in which to handle HTTP response codes of 500 (internal error), 502 (Service Temporarily Overloaded), and 503 (Service Unavailable) are received, or when no response code is received within an anticipated time.

3.3 Current Situation

Today, when a Receiver gets an HTTP error code of 502 or 503, it is free to discard the message, or do some action based on local policy. When a Sender receives these error codes, it may do any of the following:

1. Pretend nothing happened.

2. Assume a permanent communication failure has occurred. Send a Notification of Failure (PIP0A1). Restart the PIP Instance after some phone calls, or after some indefinite period.

3. Assume a temporary failure has happened, and initiate a retry algorithm.

Reaction (1) above is not appropriate.

Reaction (2) is not always correct, since a "busy" signal is usually a temporary phenomenon. Restarting the PIP Instance is not quite necessary, and is a waste of resources and time.

Reaction (3) is not very efficient in that the retry algorithm requires a retry only every 2 hrs. Usually, the "busy" situation gets better in a few minutes. Additionally, the retry algorithm is not exactly suitable for resolving HTTP issues, since it is aimed at resolving issues such as non-receipt of a Receipt-Acknowledgement or RNIF Exception messages.

However, a standardized algorithm to resolve HTTP busy messages (error codes 502 and 503) that can execute a few times within a retry window may considerably improve the scenario described above.

3.4 Detecting Permanent Failure

At the outset, the communicating partner may not know whether the failure in communication is permanent or only temporary. Detecting whether a failure in communication is permanent or temporary is an important step in recovering from a failure.

A *permanent failure* occurs when the current PIP transaction needs to be aborted, and communication with the trading partner in other ways than using the existing HTTP connection is required to resolve the problem.

All other problems are considered temporary. This section describes how to identify permanent failures from temporary failures.

Sometimes, it is not possible to classify a failure as permanent until some recovery strategy is attempted.

3.4.1 HTTP Failures

The failure classification for HTTP applies to only asynchronous RNIF requests. In the case of synchronous requests, resending the message is recommended.

Received HTTP Response Code	Classification & Action
500 Internal Error	A permanent failure. Local policy at the receiver dictates action.
502 Service Temporarily Overloaded - Server congestion; too many connections; high traffic.	A temporary failure. See "Section 3.4, Reaction to Temporary Failure" below.

503 Service Unavailable - Server busy, site may have moved, you lost your internet connection or the quality your internet connection is poor at the time.	A temporary failure. See "Section 3.4, Reaction to Temporary Failure" below.

3.4.2 RNIF Message Failures

Even when HTTP layer does not indicate any errors, RNIF message failures can still occur. These RNIF message failures and recovery methods are not specific to underlying protocol. An RNIF message failure occurs in any of the following cases:

1. No signal arrives from a PIP Responder. Obviously, this event implies the lack of receipt of an HTTP response as well.

2. An RNIF Exception signal arrives from a PIP Responder.

We consider both of these cases below in some more detail.

3.4.2.1 No Signal

Section 2.6.4.1 of RNIF 2.0 describes how to handle a case when a Receipt-Acknowledgement is not received. When no signals (Receipt-Acknowledgement or Exception) or action messages are received after the Time-to-Acknowledge * (Retry Count + 1)[1], then a permanent failure has occurred. In this case, a retry message may be sent following the guidelines in Section 2.6.4 through Section 2.6.8 of RNIF 2.0.

3.4.2.2 Exception Signal

A receipt of an exception signal is considered a temporary failure of the RNIF messaging layer. In this case, the PIP Instance may be aborted following the guidelines in [RNIF20].

3.4.3 RNIF Implementation Failure

When an RNIF implementation fails at TP B while executing a PIP Instance with TP A, it is manifested as an RNIF message failure to TP A. Such a failure may be perceived as temporary or permanent by TP A.

[1] "*" symbol stands for multiplication

3.5 Recovery Strategy

Once a failure is identified as temporary or permanent, it is possible to go to the next step, recovering from the failure (see Figure 1 below). A temporary failure is an intermittent failure.

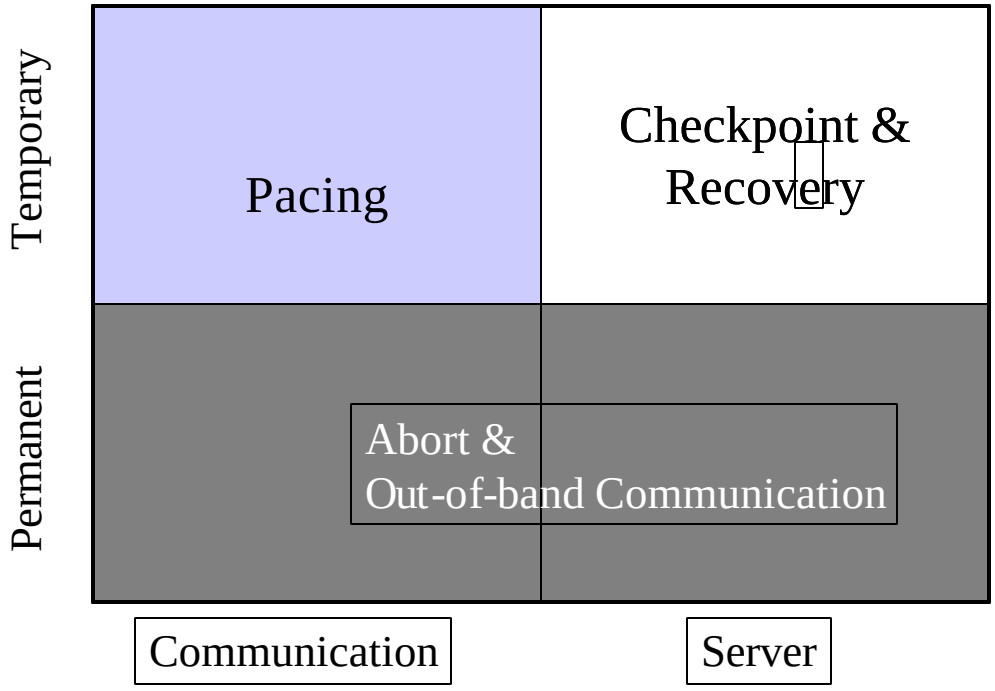

Figure 1: Strategies for Recovery

When a failure is detected, then the communicating partner may react to it in a variety of ways. If the failure is temporary, retries could be attempted or recovery can be attempted by saving contexts at checkpoints in the PIP Instance execution. If the failure is permanent, actions can be taken based on local policy. We go into details of reacting to failure presently.

3.6 Reaction to Temporary Failure

3.6.1 HTTP Error Codes 502 and 503

When the above error codes are received by TP A from TP B, the following approach is recommended. The context for this recovery approach is the following: When either of these error codes is received at TP A, it is quite possible that the HTTP server of TP B is overloaded, or somehow not able to react. In such a situation, TP A should reduce the frequency of messages to TP B so as to not overload the server at TP B any further. At the same time, TP A would want to monitor the situation at TP B, and therefore should continue to resend messages. The concepts of Pacing Interval and Pacing Count are useful here.

Pacing Interval: The time period between two consecutive resends of the action message during the Pacing Algorithm execution.

Pace Count: A Pace Count defines the maximum number of times a message will be resent to the TP A during Pacing Algorithm execution.

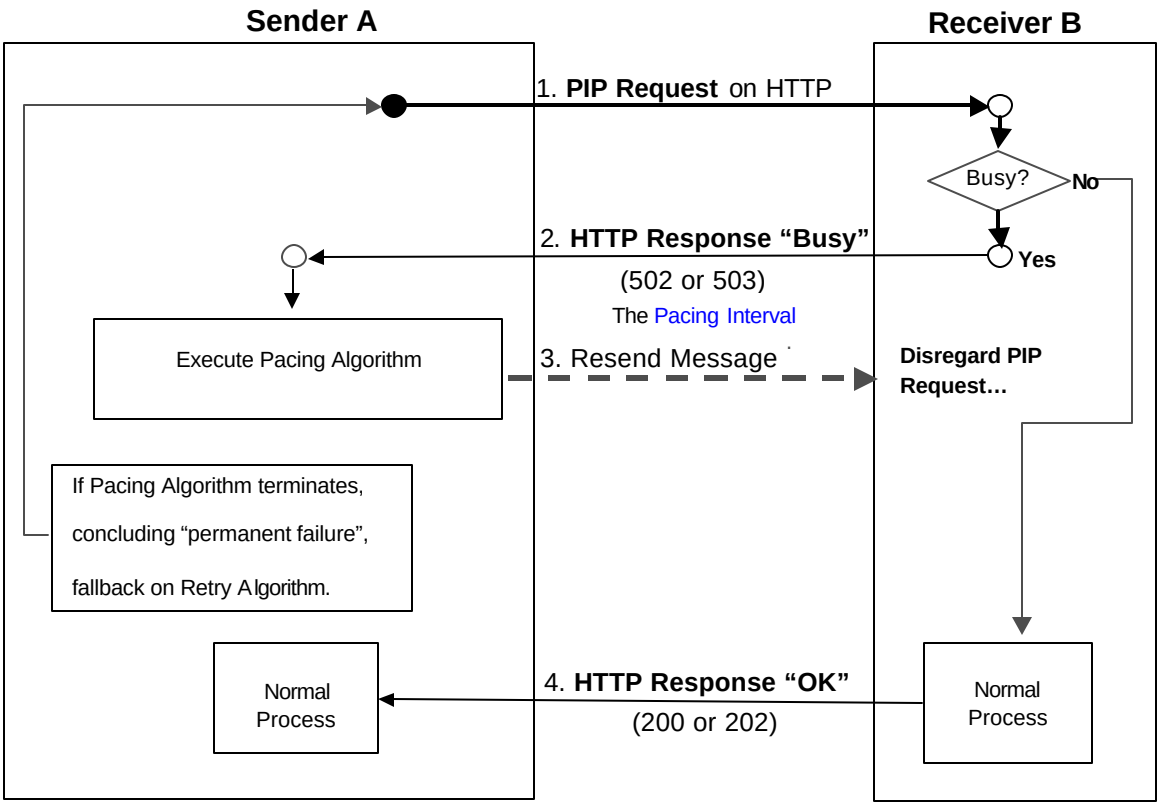

Figure 2: Pacing Algorithm

The Pacing Algorithm below applies for HTTP based asynchronous request and response.

Pacing Algorithm

Preconditions:

1. TP A has defined Pacing Interval and Pace Count
2. TP A receives HTTP error code 502 or 503 from TP B.
3. TP A is the Sender and TP B is the Receiver.

Steps:

1. TP A resends the message (action or signal) Pace Count times, every such resend happening after the elapse of a Pacing Interval. Thus, the last message will be resent at (Pace Interval * Pace Count) period, after the initial receipt of the error codes at TP A, during this instance of the Pacing Algorithm.

2. If TP A continues to receive either of the HTTP error codes 502 or 503 for all resends during this instance of the Pacing Algorithm, or receives no responses to the resends, TP A concludes that TP B has a permanent failure. On the other hand, TP B may respond with a normal reaction (a Receipt Acknowledge, Exception, or a Response message). In either case, Pacing Algorithm concludes.

3. If a Notification of Failure (PIP 0A1) arrives from TP B during this instance of Pacing Algorithm, Pacing Algorithm concludes. Pacing Algorithm will not be applied for sending PIP 0A1, since this will cause a vicious recursion.

4. While this algorithm is executing, no new PIP Instances must be initiated by TP A to send to TP B. However, Response Action messages and signals may be sent to TP B.

Post Condition

1. TP A concludes TP B has a communication failure, and whether it is permanent or temporary, OR
2. TP A concludes TP B has no communication failure (with a signal or PIP 0A1)

Notes

1. It is possible that both TP A and TP B are executing a Pacing Algorithm at the same time. Since a Pacing Algorithm is guaranteed to conclude, the end result can be either a successful conclusion of the PIP Instance, or a failure at one or both TP from the other TP's perspective.

2. If the Pacing Algorithm concludes a permanent communication failure, retry based on Retry message is to be started 2hrs after the send prior to invoking Pacing Algorithm.

3. Pacing Interval and Pacing Count must be agreed between partners.

4. Pacing Algorithm is applied only for Asynchronous (HTTP) Messages.

3.6.2 Retry Algorithm and Pacing Algorithm

The RNIF Retry Algorithm and the Pacing Algorithm described below are different, while they are related. The important differences are below:

- Retry Algorithm is not intended for HTTP failures, rather for non-receipt of Receipt Acknowledgement or RNIF Exceptions. Pacing Algorithm is intended for HTTP failures.
- When the HTTP Server is busy, or unavailable, no Receipt Acknowledgement can be sent. This means when this HTTP failure occurs, the Retry Algorithm is not executed until after Time-to-Acknowledgement is complete.
- The Retry algorithm is defined only for Action messages. The Pacing Algorithm is for both Signal and Action Messages.

The Pacing Interval is different from the "Time-to-Acknowledge interval" used with Retry Algorithm, though the concept is similar. The Pacing Interval is considerably less (typically 5 minutes), than the Time-to-Acknowledge, which is 2 hours. The Retry Count is 3, whereas, the Pacing Count can be 10. The relationship between the RNIF Retry Algorithm and HTTP Pacing Algorithm is the Pacing Algorithm is executed between two consecutive retry of action messages, or after the last retry of the action message, as per the RNIF retry algorithm. The following must hold in all situations:

Pacing Interval * (Pacing Count +1) < Time-to-Acknowledge.

3.6.3 RNIF Implementation Failure

The recovery approach may differ based upon when the failed RNIF implementation is restarted. There are two cases:

1. RNIF Implementation at TP A restarts before TP B concludes TP A has a permanent failure. In this case, if TP A had stored a PIP Instance context after its last received message, it could respond to the resends from TP B as if no failure happened. If TP A did not store such contexts, it may send a PIP 0A1 to TP B, thus aborting the current PIP Instance.

2. RNIF Implementation at TP A restarts after TP B concludes TP A has a permanent failure. In this case, TP B must have already sent a PIP 0A1 to TP A to abort the current PIP Instance. The next logical step for TP A is to abort the PIP Instance.

3.7 Reaction to Permanent Failure

When a permanent failure is detected at TP A by TP B, TP B MUST send a PIP 0A1 to TP A to abort the current PIP Instance, according to the guidelines in Section 2.6.4 to Section 2.6.8 of RNIF 2.0. However, there is no guarantee that PIP 0A1 will reach TP A. Additional responses to the situation by TP B is based on the local policy at TP B. There are varieties of local policies possible. Here is an incomplete list of possibilities:

1. Contact the failing TP through other means such as email, phone, fax, to fix the problem.

2. Wait for some time (as specified in the local policy), and check the failure has gone away using the Pacing Algorithm.

3. The Notification of Failure (PIP 0A1, phone, fax,…) must be processed by the receiver to be successful – just receiving is not enough!

3.8 Using Notification of Failure (PIP0A1)

This TR has identified when permanent failure occurs. Section 2.6.8 of the RNIF 2.0 specification provides additional guidance on sending PIP 0A1.

Given the importance of PIP 0A1, we recommend the following:

1. The system readiness to receive PIP 0A1 is required. (RNIF specification allows any partners to send PIP 0A1 to any partners.)

2. The sender of PIP 0A1 is responsible for resolving errors occurring from aborting a PIP Instance at its site. It is worthwhile preparing for manual operations procedures for this purpose.

3. It is also helpful to have prior agreement with trading partners on when PIP 0A1 will be used.

4 Benefits

Using this high availability technique described in this TR has the following benefits:

1. One TP is able to determine whether the other TP has a permanent failure or has shut down. This results in more effective communication with the failing partner through other means, as well as, efficient failure monitoring with reduced messages to the failing partner.

2. Recovery in some intermittent failure situations is possible, and the appropriate way to do that is defined in this TR.

Note that the techniques defined in this TR, while useful in the scenarios described, may also work for other failure scenarios. However, the scope of the TR doesn't cover those additional failure scenarios.

5 Implementation Considerations

1. It is advised that the Time to Acknowledge is at least greater than the Pacing Interval for the corresponding action message. It would be advisable for the Time to Acknowledge to be several times the Pacing Interval.

2. The least value of Pacing Interval and the maximum value of Pace Count may need to be agreed upon by the partners for communication in both directions. These values may be different for different TPs because each partner may have different system capacity.

3. When the Pacing Algorithm is being executed, if any new PIP Instances were initiated, the resulting messages must not be sent to the trading partner. However, such messages may be queued at the trading partner for future transmission at the favorable conclusion of the Pacing Algorithm.

4. The Pacing Algorithm may be implemented as described in the figure below, as part of RNIF implementation.

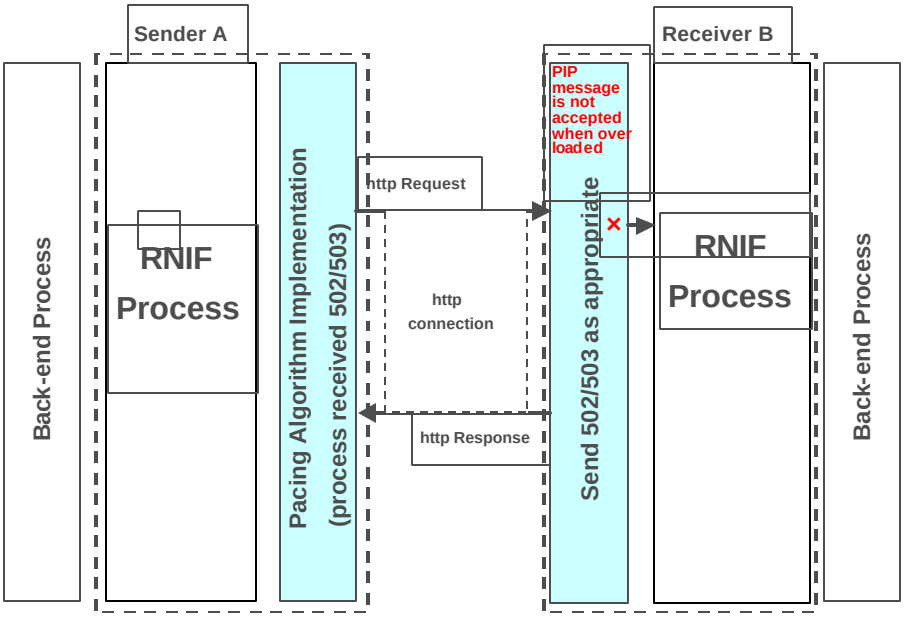

Figure 3: Pacing Algorithm Implementation

5. Note that there is an implicit assumption in the Pacing Algorithm that the Pacing Interval is the same between the resending of messages. This assumption is not necessarily true in some implementations. In such cases, the constraint

   Pacing Interval * (Pacing Count +1) < Time-to-Acknowledge

   will need to be redone so that the sum of all the Pacing Intervals is less than Time-to-Acknowledge.

6. While this Technical Recommendation specifically applies to RNIF 2.0, a Solution Provider or an Implementer may implement this for RNIF 1.1, as well.

6 References

[RNIF20] “RosettaNet Implementation Framework: Core Specification,” Version V02.00.01, March 6, 2002.

