

BONDI

BONDI ARCHITECTURE & SECURITY REQUIREMENTS (APPENDICES)

VERSION:	Version 1. 1
STATUS:	Approved Release
DATE OF LAST EDIT:	29 January 2010
OWNER	OMTP Limited

CONTENTS

APPENDIX A. BONDI-DEFINED FEATURES	6
A.1 INTRODUCTION	6
A.2 BONDI PROCESSING ASSERTIONS.....	6
A.2.1 <i>UI presentation</i>	6
A.2.2 <i>Widget Persistent Storage</i>	8
A.2.3 <i>Network utilisation</i>	8
A.3 WIDGET APPLICATION MANAGEMENT	9
A.3.1 <i>LyfeCycle</i>	9
A.4 NETWORK ACCESS FEATURES.....	10
APPENDIX B. SECURITY MODEL DEFINITION	12
B.1 INTRODUCTION	12
B.2 APPLICATION EXECUTION PHASES	12
B.3 VALUES AND TYPES	12
B.4 SUBJECT ATTRIBUTES.....	13
B.4.1 <i>Widget Resource Identity</i>	14
B.4.2 <i>Website Identity</i>	15
B.5 RESOURCE ATTRIBUTES	16
B.6 ENVIRONMENT ATTRIBUTES.....	17
B.7 ATTRIBUTE MATCH	17
B.8 SUBJECT SPECIFICATION.....	18
B.9 TARGET	18
B.10 DECISION	19
B.11 RULE	19
B.12 CONDITION.....	19
B.13 POLICY	19
B.14 POLICY SET.....	20
B.15 POLICY DOCUMENT	20
B.16 SIGNED POLICY DOCUMENT	20
B.17 MATCHING FUNCTION	20
B.17.1 <i>String Equality Matching Function</i>	21

<i>B.17.2</i>	<i> Globbing Matching Function.....</i>	<i>21</i>
<i>B.17.3</i>	<i> Regular Expression Matching Function.....</i>	<i>21</i>
B.18	MODIFIER FUNCTION.....	21
<i>B.18.1</i>	<i> URI-Scheme Modifier Function.....</i>	<i>21</i>
<i>B.18.2</i>	<i> URI-Authority Modifier Function.....</i>	<i>22</i>
<i>B.18.3</i>	<i> URI-Scheme-Authority Modifier Function.....</i>	<i>22</i>
<i>B.18.4</i>	<i> URI-Host Modifier Function.....</i>	<i>22</i>
<i>B.18.5</i>	<i> URI-Path Modifier Function.....</i>	<i>22</i>
B.19	COMBINING ALGORITHM.....	23
<i>B.19.1</i>	<i> Deny-Overrides Combining Algorithm.....</i>	<i>23</i>
<i>B.19.2</i>	<i> Permit-Overrides Combining Algorithm.....</i>	<i>23</i>
<i>B.19.3</i>	<i> First-Applicable Rule Combining Algorithm.....</i>	<i>24</i>
<i>B.19.4</i>	<i> First-Matching-Target Policy Combining Algorithm.....</i>	<i>24</i>
B.20	EFFECT.....	25
<i>B.20.1</i>	<i> Permit.....</i>	<i>25</i>
<i>B.20.2</i>	<i> Deny.....</i>	<i>25</i>
<i>B.20.3</i>	<i> prompt-x.....</i>	<i>25</i>
B.21	QUERY.....	25
APPENDIX C.	SECURITY POLICY DOCUMENT FORMAT.....	27
C.1	INTRODUCTION.....	27
C.2	SCHEMA.....	27
<i>C.2.1</i>	<i> <signed-policy>.....</i>	<i>27</i>
<i>C.2.2</i>	<i> <signature>.....</i>	<i>27</i>
<i>C.2.3</i>	<i> <policy-set>.....</i>	<i>27</i>
<i>C.2.4</i>	<i> <policy>.....</i>	<i>28</i>
<i>C.2.5</i>	<i> <rule>.....</i>	<i>28</i>
<i>C.2.6</i>	<i> <target>.....</i>	<i>28</i>
<i>C.2.7</i>	<i> <subject>.....</i>	<i>28</i>
<i>C.2.8</i>	<i> <condition>.....</i>	<i>28</i>
<i>C.2.9</i>	<i> <subject-match>, <resource-match>, <environment-match>....</i>	<i>29</i>
<i>C.2.10</i>	<i> <subject-attr>, <resource-attr>, <environment-attr>.....</i>	<i>29</i>

The information contained in this document represents the current view held by OMTP Ltd. on the issues discussed as of the date of publication.

This document is provided “as is” with no warranties whatsoever including any warranty of merchantability, non-infringement, or fitness for any particular purpose. All liability (including liability for infringement of any property rights) relating to the use of information in this document is disclaimed. No license, express or implied, to any intellectual property rights are granted herein.

This document is distributed for informational purposes only and is subject to change without notice. Readers should not design products based solely on this document.

© 2009-2010 OMTP Ltd. All rights reserved. OMTP and OMTP BONDI are registered trademarks of OMTP Ltd.

Document History

VERSION	DATE	EDITOR	REMARKS
1.01	27/07/09	OMTP	Errata: <ul style="list-style-type: none"> • Updates to api-feature and device-cap • Conversion of IRIs to URIs where required Minor editorials • Updates to references
1.1	28/08/09	Paddy Byers	First set of updates for v1.1.
1.1	4/9/09	Paddy Byers	Further updates to address: <ul style="list-style-type: none"> - requirements for termination of running Web Applications in the event of a policy update; - requirements for access control in child browsing contexts.
1.1	21/9/09	Paddy Byers	Updates to reflect agreed actions following group review
1.1	29/1/10	Paddy Byers	Updates following feedback on CR release

Appendix A. BONDI-DEFINED FEATURES

A.1 INTRODUCTION

This section is non-normative.

BONDI defines a series of Features that a Web Application can express a dependency on, and a series of Device Capabilities associated with the corresponding underlying functionality of the platform. Features and Device Capabilities associated with the BONDI-defined APIs are defined formally as part of the definition of each of the BONDI interfaces¹.

Additionally, BONDI defines Features and Device Capabilities that are not associated with JavaScript APIs:

- **BONDI processing assertions:** Features representing metadata and directives relate to the processing of a Web Application.
- **Widget Network Access Features:** Features that make use of network IO and are subject to BONDI access controls but are not associated with BONDI JavaScript APIs.
- **Widget application management:** Features that represent application management actions on the widget.

These are defined in this appendix.

A.2 BONDI PROCESSING ASSERTIONS

This section is normative.

A.2.1 UI PRESENTATION

Base IRI:

`http://bondi.omtp.org/meta/ui`

The `ui` Feature is used to provide information about the way in which the Widget may behave in the user interface.

This element governs the following two aspects of behaviour:

- **background operation.** The presentation and organisation of running Widgets in a terminal user interface (UI) depends on the nature of the terminal and its UI and whether, for example, it supports multiple

¹ http://bondi.omtp.org/1.0/apis/BONDI_Interface_Requirements_v1.0.pdf

concurrently running applications in independent windows or a single application is visible in the foreground at any given time. However, when an application is not in the foreground it may be hidden and the Web Runtime is usually entitled to suspend delivery and processing of events for that application, or deny it access to other system resources. If a Web Application requires that this does not happen – for example a music player Widget would require that audio output continues even when in the background – it must request this explicitly.

- **hidden operation.** A Widget might have no requirement to be displayed or to interact directly with the user – for example it might solely be required to respond when system events (such as incoming messages) are directed to it – and it is possible then for the Widget to have no associated window or any visible representation in the UI. Since the user might not be able to navigate to that running Widget instance and terminate it, there is an extra obligation on the Widget author to ensure that the Widget correctly handles all lifecycle events appropriately and does not behave in an undesirable way whilst being invisible to the user.

Each of these specific dependencies is expressed as a CGI-encoded parameter to the base IRI as follows.

`background`

The Widget requires background operation.

Use: Optional

Allowable values: `true, false`

Default value: `false`

`hidden`

The Widget requires hidden operation.

Use: Optional

Allowable values: `true, false`

Default value: `false`

If any of the parameters of the `ui` Feature are omitted the Web Runtime SHALL assume the default value for that parameter.

Example:

`http://bondi.omtp.org/meta/ui?background=true&hidden=false`

A.2.2 **WIDGET PERSISTENT STORAGE**

Base IRI:

`http://bondi.omtp.org/meta/storage`

The `storage` Feature is used to provide information about the resource requirements of the Widget in respect of persistent storage. At this release, parameters are defined relating only to *private* widget storage which is accessible only to instances of that Widget.

The storage requirement is expressed as a CGI-encoded parameter to the base IRI as follows.

`min-private`

The minimum space required by the Widget for private persistent storage.

Use: Optional

Allowable values: non-negative decimal integer string plus SI/IEC unit string

Default value: no private persistent storage required.

Example:

<http://bondi.omtp.org/meta/storage?min-private=50KiB>

A.2.3 **NETWORK UTILISATION**

Base IRI:

`http://bondi.omtp.org/meta/network`

The `network` Feature is used to provide information about the way in which the Widget makes use of network resources.

Each of these specific declarations is expressed as a CGI-encoded parameter to the base IRI as follows.

`autonomous`

Indicates whether or not the app will make connections to the network autonomously, instead of solely in response to user input.

Use: Optional

Allowable values: `true`, `false`

Default value: `false`

`interval`

Indicates the typical interval between autonomous network connection attempts.

Use: Optional

Allowable values: non-negative decimal integer string plus time interval unit string [ms|s|m|h|d]

Default value: if this parameter is not specified, the interval is unknown.

`min-volume`

Indicates minimum required aggregate upload and download data transfer rate.

Use: Optional

Allowable values: non-negative decimal integer string plus SI/IEC unit string

Default value: if this parameter is not specified, the transfer rate is unknown.

Examples:

<http://bondi.omtp.org/meta/network?autonomous=true&interval=1h>

<http://bondi.omtp.org/meta/network?min-volume=1KiB/h>

A.3 WIDGET APPLICATION MANAGEMENT

This section identifies Features associated with application management actions performed by the Widget Manager.

These features can be used in a Security Policy to exercise explicit control over the installation and instantiation of Widgets. These Features cannot be dependencies of any specific Widget and it is not required to declare these features in a Widget Configuration Document.

A.3.1 LIFECYCLE

Base IRI:

`http://bondi.omtp.org/lifecycle`

The `lifecycle` Features identify those Widget lifecycle actions that are mediated by Security Policy.

Each of these specific Features is defined below.

`http://bondi.omtp.org/lifecycle/widget-install`

Corresponds to Widget installation. Prior to installation of a Widget Resource, a Web Runtime MUST evaluate an access control Query, with an `api-feature` resource attribute equal to the above IRI, against the configured Security Policy, to determine whether or not installation is permitted.

`http://bondi.omtp.org/lifecycle/widget-instantiate`

Corresponds to Widget instantiation. Prior to instantiation of a Widget, a Web Runtime MUST evaluate an access control Query, with an `api-feature` resource attribute equal to the above IRI, against the configured Security Policy, to determine whether or not instantiation is permitted.

`http://bondi.omtp.org/lifecycle/widget-update`

Corresponds to Widget update. Prior to update of a Widget Resource, a Web Runtime MUST evaluate an access control Query, with an `api-feature` resource attribute equal to the above IRI, against the configured Security Policy, to determine whether or not update is permitted.

`http://bondi.omtp.org/lifecycle/widget-uninstall`

Corresponds to Widget removal. Prior to removal of a Widget, a Web Runtime MUST evaluate an access control Query, with an `api-feature` resource attribute equal to the above IRI, against the configured Security Policy, to determine whether or not removal is permitted.

A.4 NETWORK ACCESS FEATURES

This section identifies mechanisms by which Widgets make use of network IO and are subject to BONDI access controls but are not associated with BONDI JavaScript APIs. These mechanisms, when used by a Widget, are associated with the following Feature:

`http://bondi.omtp.org/api/bondi.networkaccess`

They use Device Capabilities (and present the associated Device Capability parameters) as indicated in the table below. (Note that use of these Features is additionally constrained by the `<network-access>` declarations made in the Widget Configuration Document.)

BONDI does not specify any access controls applicable to these Features when used from a Website.

NETWORK ACCESS MECHANISM	USES DEVICE CAPABILITY	DEVICE CAPABILITY PARAMETERS
XMLHttpRequest	io.http.client (for HTTP) or io.https.client (for HTTPS)	<code>uri</code> contains the second argument to <code>open()</code>
External network access via HTML elements with <code>src</code> references to external sites, and other means.	io.http.client (for HTTP) or io.https.client (for HTTPS)	<code>uri</code> contains the value of the <code>src</code> attribute

Appendix B. SECURITY MODEL DEFINITION

This section is normative.

B.1 INTRODUCTION

This appendix defines the formal model underlying the general security framework. This includes definitions of each of the entities involved in the definition of an access control policy, and a definition of the attributes of each entity that are recognised and are required to be supported.

B.2 APPLICATION EXECUTION PHASES

The *execution phase* of a Web Application reflects the state of that application at the time an associated access control query is made. The defined execution phases are listed below.

EXECUTION PHASE	DESCRIPTION
widget-install	Applies to access control queries made by a Widget User Agent during the processing of a Widget Resource as part of an installation or update operation.
widget-instantiate	Applies to access control queries made by a Widget User Agent during the instantiation of a Widget.
website-bind	Applies to access control queries made in response to a call to requestFeature() in the course of execution of a Website
invoke	Applies to access control queries made in response to invocation of a JavaScript API in the course of execution of a Web Application

B.3 VALUES AND TYPES

Each value in an expression is conceptually a *bag* of potentially multiple simple values. The bag can be empty, containing no simple values. In practice almost every value encountered in the model is either an empty bag or a bag containing a single simple value. When a bag contains one or more simple values, all the simple values have the same type, one of:

- string;
- IRI.

Each modifier function (section B.18) defines its result type, and how the function's effect depends on the type of the input.

Each matching function (section B.15) defines how it depends on the type of its input.

Where a modifier function or matching function does not specify how it treats an input of a particular type, it implicitly converts the value to a bag of strings before performing its operation.

When evaluating an access control query at a given application Execution Phase, an expression may have undetermined value if one or more of the attributes on which it depends has undetermined value at that execution phase.

For each modifier function (section B.18) and matching function (section B.15), its result for a given set of inputs is determined if and only if all of its inputs are determined.

The syntax used for encoding a certificate fingerprint in BONDI Security Policy documents is the SDP syntax defined in RFC 4572² without the "fingerprint" scheme, as follows:

```
bondifingerprint = hash-func SP fingerprint
hash-func       = "sha-1" / "sha-224" / "sha-256" /
                  "sha-384" / "sha-512" /
                  "md5" / "md2" / token
                  ; Additional hash functions can only come
                  ; from updates to RFC 3279
Fingerprint     = 2UHEX *(":" 2UHEX)
                  ; Each byte in upper-case hex, separated
                  ; by colons.
UHEX            = DIGIT / %x41-46 ; A-F uppercase
```

Note: RFC 3279³

B.4 SUBJECT ATTRIBUTES

A subject corresponds to an entity that may attempt security-relevant actions and corresponds to a single "identity". (In practice, some Web Applications might have multiple identities – for example is a Widget Resource is signed by multiple signers – but for the purposes of this model, each access control query is considered to involve a single subject and hence a single identity.)

² <http://www.ietf.org/rfc/rfc4572.txt>

³ <http://www.ietf.org/rfc/rfc3279.txt>

The identity of a *subject* is in one of the following classes. The class determines which attributes are available; other attributes have the undefined value.

All subject attributes are determined for all applicable application Execution Phases.

B.4.1 WIDGET RESOURCE IDENTITY

The Widget identity type applies to all operations associated with a Widget Resource, or occurring in the execution of a document belonging to a Widget Resource.

Operations occurring in the execution of a remotely hosted document that has been loaded by a Widget (for example in an iframe) use a Website identity (see B.4.2 below).

ATTRIBUTE	TYPE	VALUE
class	String	This has the value "widget" if and only if the subject is a Widget.
install-uri	URI	The URI that the Widget Resource was originally retrieved from before installation, if known, otherwise the empty bag.
id	URI	The identity of the Widget. For a W3C Widget specification compliant Widget Resource, this is the value of the id attribute of the <widget> element in the Widget Configuration Document converted from IRI to URI based on RFC3987. In this case, it is a URI that uniquely identifies the Widget. Empty bag if there is no id attribute.
version	string	Version of the Widget Resource. For a W3C Widget specification compliant Widget Resource, this is the version attribute of the <widget> element in the Widget Configuration Document. Empty bag if there is no version attribute.
distributor-key-cn	string	The common name of the end entity certificate for the applicable Widget Resource distributor signature. Empty bag if none.
distributor-key -fingerprint	string	The fingerprint of the end-entity certificate for the applicable Widget Resource distributor signature. Empty bag if none.
distributor-key-root-cn	string	The common name of the root certificate for the applicable Widget Resource distributor signature. Empty bag if none.
distributor-key-root-fingerprint	string	The fingerprint of the root certificate for the applicable Widget Resource distributor signature. Empty bag if none.
author-key-cn	string	The common name of the end entity certificate for the Widget Resource author signature. Empty bag if none.
author-key -fingerprint	string	The fingerprint of the end entity certificate for the Widget Resource author signature in SDP syntax. Empty bag if none.
author-key-root-cn	string	The common name of the root certificate for the Widget Resource author signature. Empty bag if none.

author-key-root-fingerprint	string	The fingerprint of the root certificate for the Widget Resource author signature.. Empty bag if none.
widget-attr:name		The value of the named attribute of the <widget> element whose type and value are set up in the Widget Configuration Document for use in the BONDI security framework. Empty bag if no such named attribute is defined.

B.4.2 WEBSITE IDENTITY

The Website identity type applies to all operations occurring in the execution of a remotely-hosted document, whether this is the top-level document of the Website or is associated with some child browsing context (such as an iframe).

ATTRIBUTE	TYPE	VALUE	MEANING
class	String	"website"	Has the value "website" if and only if the subject is of this class.
sign-schema	string	"" (empty string)	Not signed.
		"tls"	The page was fetched using HTTPS and the browser has verified that the site certificate's Common Name matches the host that the page was fetched from, and it has already applied its own policies regarding whether the root certificate is in an acceptable trust domain.
		"tls-ev"	As "tls", and, additionally, the site certificate has an extended validation field and the browser's internal policy allows that information to be passed to the BONDI security framework.
uri	URI		The URI used to access the document that embeds or refers to the JavaScript code, corresponding to the window.location property of the browsing context. In the case of that a Feature is accessed from a child browsing context (for example from within a <iframe> within some outer document), this attribute provides the location of the child context.
uri-top	URI		The URI used to access the Website that embeds or refers to the JavaScript code, corresponding to the top.window property of the browsing context. In the case that the Feature is accessed from a child browsing context (for example from within an <iframe>), this attribute provides the location of the top-level browsing context. If the current browsing context is a child of a Widget top-level browsing context, this attribute contains an IRI with the widget: scheme that corresponds to the top-

		level containing document from the Widget Resource.
key-root-cn	string	The common name of the root certificate chained to by the site certificate. Empty bag if none.
key-root-fingerprint	string	The fingerprint of the root certificate chained to by the site certificate. . Empty bag if none.

B.5 RESOURCE ATTRIBUTES

The *resource* is identified by one or more of the following attributes:

ATTRIBUTE	TYPE	VALUE	COMMENT
api-feature ⁴	URI	The IRI identifier of the requested Feature converted to URI as per RFC3987.	This uses the same naming scheme as in a widget's <feature> element. See Appendix A. Determined for all applicable application Execution Phases.
device-cap	string	Device capability being accessed, if any. Empty bag if none.	See Appendix A. Determined for all applicable application Execution Phases.
param:name	See comment	The value of parameter name.	The specification of each Device Capabilities lists the parameters associated with that Device Capability and the type and semantics of each. Empty bag if the parameter is not defined. Determined in the invoke execution phase. Undetermined in all other execution phases.
The following resource attributes give information on the source of the implementation of the API Feature.			
feature-install-uri	URI	The URI that the API implementation was originally retrieved from before installation, if known, otherwise the empty bag.	Determined for all applicable application Execution Phases.
feature-key-cn	string	The common name of the end entity certificate for the signature associated with the Feature implementation. Empty bag if none.	Determined for all applicable application Execution Phases.
feature-key-root-cn	string	The common name of the root certificate for the signature associated with the Feature implementation. Empty bag if none.	Determined for all applicable application Execution Phases.
feature-key-root-fingerprint	string	The fingerprint of the root certificate of the signature associated with the Feature implementation. Empty bag	Determined for all applicable application Execution Phases.

⁴ Note that this Resource Attribute is presented for every Feature request, whether associated with a JavaScript API or not.

		if none.	
--	--	----------	--

B.6 ENVIRONMENT ATTRIBUTES

Attributes of the *environment* capture contextual information relating to the device or other circumstances of the access attempt.

ATTRIBUTE	TYPE	VALUE	COMMENT
roaming	string	"national", "international", or empty string	Determined in the following Execution Phases: - widget-instantiate - website-bind - invoke Undetermined in the following Execution Phases: - widget-install
bearer-type	String	The type of the current network bearer over which a network request will be served, either by request of the application or by default (per the current serving network or the one over which the request will be served, if multiple networks are available). A comma-separated list of one or more of the bearer types given as examples in W3C DCO ⁵ .	Determined in the following Execution Phases: - widget-instantiate - website-bind - invoke Undetermined in the following Execution Phases: - widget-install

B.7 ATTRIBUTE MATCH

An attribute match is a statement about one attribute whose truth can be evaluated, that is it evaluates to true or false (or undetermined). An attribute match is a subject match, resource match or environment match, depending on whether the attribute being matched is a subject, resource or environment attribute.

An attribute match is an expression with a boolean result whose form is limited to one of the following:

- matchfunc(modifierfunc(attr), value)
- matchfunc(attr, value)

Matchfunc is the matching function, a function with a boolean result and two non-boolean inputs. Its result is undetermined if either input is undetermined.

In the first case, modifierfunc is a function with a non-boolean result and a single non-boolean input. The result of modifierfunc is undetermined if its input is undetermined.

⁵ <http://www.w3.org/TR/dcontology/#BearerType>

In the second case, there is no modifierfunc.

The value to match (matchfunc's second input) is a sequence of literal text and other attribute references implicitly combined using string concatenation. Thus its type is bag containing a single string, unless there is any reference to an attribute resolving to an empty bag, in which case it is an empty bag. Any reference to a non-string attribute is converted to string bag first. Any reference to an attribute whose value is a bag containing two or more values causes the whole match value to be undefined. Any reference to an undetermined attribute causes the whole value to match to be undetermined.

For a subject attribute match, only a single literal string is allowed, with no attribute references.

If the attribute does not exist, then it has the empty bag value.

B.8 SUBJECT SPECIFICATION

A *subject* specification consists of a conjunctive sequence of *subject* matches.

A subject specification is evaluated as follows:

- is determined and has value TRUE if each of the *subject* matches has value TRUE
- otherwise, is undetermined if any or the *subject* matches is undetermined
- otherwise is determined and has value FALSE.

A *subject* match is an attribute match where the attribute being matched is a *subject* attribute, and the match value is a literal string and does not contain any attribute references.

B.9 TARGET

The *target* of a *policy* or *policy set* identifies the set of *subjects* to which the *policy* or *policy set* applies.

The *target* consists of a disjunctive sequence of *subject* specifications.

A target specification is evaluated as follows:

- has value TRUE if at least one of the *subject* specifications has value TRUE
- otherwise has value FALSE.

A *policy* or *policy-set* that has no *target* explicitly specified is treated as having a *target* that evaluates unconditionally to TRUE.

B.10 DECISION

If determined, the result of a *rule* or *policy* or *policy set* is a *decision*, either “not applicable” or any one of the *effects* “permit”, “prompt-blanket”, “prompt-session”, “prompt-oneshot” or “deny”. The *effects* are defined in section B.20.

The result of a *rule* or *policy* or *policy set* may be undetermined under conditions specified for each below.

B.11 RULE

A *rule* consists of a *condition* and an *effect*. The possible *effects* are defined in section B.20.

The result of a rule is determined if and only if its condition has determined value.

B.12 CONDITION

The *condition* of a *rule* specifies extra criteria that need to be matched before the *rule* becomes applicable.

The *condition* consists of one or more attribute matches, combined with AND and OR operators into an arbitrarily nested tree.

The AND operator is evaluated as follows:

- is determined and has value “no match” if any input is “no match”;
- otherwise is undetermined if any input is undetermined;
- otherwise is determined and has value “match”.

The OR operator is evaluated as follows:

- is determined and has value “match” if any input is “match”;
- otherwise is undetermined if any input is undetermined;
- otherwise is determined and has value “no match”.

B.13 POLICY

A *policy* has a *target*, and a list of zero or more *rules* combined using a *rule-combining algorithm*. See section B.19 for the combining algorithms. Where a

directive attribute query finds more than one applicable directive attribute set, the first one is used.

A *policy* optionally has a textual description.

A *policy* optionally has an id. If an implementation provides a means to provision a security policy fragment to replace an existing one, this id can be used to identify the *policy* or *policy set* to replace. No management of ids is mandated, therefore it is recommended that a standardised textual representation of a UUID should be used as the id.

The result of a policy is determined if and only if its combining rule has determined value.

B.14 POLICY SET

The overall security framework is a *policy set*.

A *policy set* is a *target* with a list of zero or more *policies* and *policy sets* combined using a *policy-combining algorithm*. See section B.19 for the combining algorithms. Where a directive attribute query finds more than one applicable directive attribute set, the first one is used.

A *policy set* optionally has an id. If an implementation provides a means to provision a security policy fragment to replace an existing one, this id can be used to identify the *policy* or *policy set* to replace. No management of ids is mandated, therefore it is recommended that a standardised textual representation of a UUID should be used as the id.

The result of a policy is determined if and only if its combining rule has determined value.

B.15 POLICY DOCUMENT

Where the implementation supports deployment of a fragment of policy to add to the existing security policy framework or to replace a part of it, the *policy document* is the unit of addition or replacement. A *policy document* can be either a *policy* or a *policy set*.

B.16 SIGNED POLICY DOCUMENT

Where the implementation supports deployment of policy fragments as above, the *signed policy document* is the cryptographically signed unit of deployment. It contains one or more *policy documents* as well as a single signature.

B.17 MATCHING FUNCTION

The matching function used in an attribute match is one of the following.

B.17.1 STRING EQUALITY MATCHING FUNCTION

True if and only if some string from one input string bag is byte-for-byte equal to some string from the other input string bag. Thus an empty bag is not equal to anything, not even another empty bag. An input of type other than empty bag or string bag is converted to string bag first.

B.17.2 GLOBBING MATCHING FUNCTION

True if and only if, for some string in the first input string bag, the entire string matches the glob pattern in some string in the second input string bag. If either input is the empty bag, the result is false. An input of type other than empty bag or string bag is converted to string bag first.

A glob pattern is as described in SUSv3⁶ section 2.13 Pattern Matching Notation but excluding 2.13.3 Patterns Used for Filename Expansion.

Using this function with a glob pattern of "*" (a single asterisk) is a convenient way to test whether the first input is not an empty bag.

B.17.3 REGULAR EXPRESSION MATCHING FUNCTION

True if and only if, for some string in the first input string bag, some part of the string matches the regular expression pattern in some string in the second input string bag. If either input is the empty bag, the result is false. An input of type other than empty bag or string bag is converted to string bag first.

This uses the definition of regular expressions in ECMAScript 3rd edition⁷.

B.18 MODIFIER FUNCTION

The modifier function optionally specified in each attribute in a target or condition is one of the following.

B.18.1 URI-SCHEME MODIFIER FUNCTION

If the input is a string bag, first it is converted to a URI bag by interpreting each string as a URI. Any string that does not have the form of a URI is removed from the bag.

Each URI in the bag is converted to a string by taking the URI's scheme component.

Thus the result type is either the empty bag or string bag.

⁶ http://www.unix.org/single_unix_specification/

⁷ <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf>

B.18.2 URI-AUTHORITY MODIFIER FUNCTION

If the input is a string bag, first it is converted to a URI bag by interpreting each string as a URI. Any string that does not have the form of a URI is removed from the bag.

Each URI in the bag is converted to a string by taking the URI's authority component. If the URI does not have an authority component, it is removed from the bag.

Thus the result type is either the empty bag or string bag.

B.18.3 URI-SCHEME-AUTHORITY MODIFIER FUNCTION

If the input is a string bag, first it is converted to a URI bag by interpreting each string as a URI. Any string that does not have the form of a URI is removed from the bag.

Each URI in the bag is converted to a string by taking the URI's scheme and authority components. If the URI does not have an authority component, it is removed from the bag.

Thus the result type is either the empty bag or string bag.

B.18.4 URI-HOST MODIFIER FUNCTION

If the input is a string bag, first it is converted to a URI bag by interpreting each string as a URI. Any string that does not have the form of a URI is removed from the bag.

Each URI in the bag is converted to a string by taking the URI's host component. If the URI does not have an authority component, it is removed from the bag.

Thus the result type is either the empty bag or string bag.

B.18.5 URI-PATH MODIFIER FUNCTION

If the input is a string bag, first it is converted to a URI bag by interpreting each string as a URI. Any string that does not have the form of a URI is removed from the bag.

Each URI in the bag is converted to a string by taking the URI's path component. If the URI does not have an authority component, it is removed from the bag.

Thus the result type is either the empty bag or string bag.

B.19 COMBINING ALGORITHM

The *policy-combining algorithm* for a *policy set* determines how child *policies* and *policy sets* are combined.

The *rule-combining algorithm* for a *policy* determines how child *rules* are combined.

The algorithms are described in the following subsections. The term *child* is used to mean the child *rules* in the *policy* when applying the *policy's rule-combining algorithm*, or the child *policies* and *policy sets* in the *policy set* when applying the *policy set's policy-combining algorithm*.

B.19.1 DENY-OVERRIDES COMBINING ALGORITHM

The Deny-Overrides Combining Algorithm is usable as a policy-combining algorithm and as a rule-combining algorithm.

The overall result of a *query* is evaluated as follows.

- If any child evaluates to "deny", then the overall result is "deny".
- Otherwise, if any child is undetermined, then the overall result is undetermined.
- Otherwise, if any child evaluates to "prompt-oneshot", then the overall result is "prompt-oneshot".
- Otherwise, if any child evaluates to "prompt-session", then the overall result is "prompt-session".
- Otherwise, if any child evaluates to "prompt-blanket", then the overall result is "prompt-blanket".
- Otherwise, if any child evaluates to "permit", then the overall result is "permit".
- Otherwise, the overall result is "inapplicable".

B.19.2 PERMIT-OVERRIDES COMBINING ALGORITHM

The Permit-Overrides Combining Algorithm is usable as a policy-combining algorithm and as a rule-combining algorithm. The overall result of a *query* is evaluated as follows.

- If any child evaluates to "permit", then the overall result is "permit".

- Otherwise, if any child is undetermined, then the overall result is undetermined.
- Otherwise, if any child evaluates to "prompt-blanket", then the overall result is "prompt-blanket".
- Otherwise, if any child evaluates to "prompt-session", then the overall result is "prompt-session".
- Otherwise, if any child evaluates to "prompt-oneshot", then the overall result is "prompt-oneshot".
- Otherwise, if any child evaluates to "deny", then the overall result is "deny".
- Otherwise, the overall result is "inapplicable".

B.19.3 FIRST-APPLICABLE RULE COMBINING ALGORITHM

The First-Applicable Rule Combining Algorithm is usable as a rule-combining algorithm.

The overall result of a query is evaluated by processing the children in written order as follows:

- if the current child is determined and does not evaluate to "inapplicable", the overall result is the result of the current child;
- otherwise, if the current child is undetermined, the overall result is undetermined;
- otherwise, if the current child is determined and has value "inapplicable", continue processing at the next child. If already processing the final child, the overall result is "inapplicable".

B.19.4 FIRST-MATCHING-TARGET POLICY COMBINING ALGORITHM

The First-Matching-Target Policy Combining Algorithm is usable as a policy-combining algorithm.

The overall result of a query is evaluated by processing the children in written order as follows:

- if the current child has a target that matches the overall result is the result of the current child;
- otherwise, continue processing at the next child. If already processing the final child, the overall result is "inapplicable".

B.20 EFFECT

The *effect* of a *rule* is one of the following:

B.20.1 PERMIT

This *effect* allows requested access without user interaction.

B.20.2 DENY

This *effect* denies requested access without user interaction.

B.20.3 PROMPT-X

The *prompt-oneshot*, *prompt-session* and *prompt-blanket effects* allow requested access after explicit confirmation by the user. The implementation **MUST** prompt the user before allowing access.

The implementation **MUST** only provide the user the option to grant permission up to the maximum allowed by the *effect*, *ie*:

- *prompt-oneshot*: "deny always", "deny this time", "allow this time";
- *prompt-session*: *prompt-oneshot* options plus "deny for this session", "allow for this session";
- *prompt-blanket*: *prompt-session* options plus "allow always".

The implementation **MUST** provide a means to respond with any available option that is applicable in the context in which the prompt is displayed.

Any default action **MUST** be at least as restrictive as "deny this time".

If the user has the option of deferring a response indefinitely and the user does not respond explicitly, the requested access **MUST NOT** be allowed.

For a *Widget*, a session lasts while the application is still running and the terminal has not been switched off or placed in standby mode.

For a *Website*, another visit to the same page in the same Browser tab or window is part of the same session.

B.21 QUERY

A *query* represents a specific instance of a security policy being evaluated in order to make an access control decision relating to an attempted operation by a *Web Application*..

A *query* is characterised by the collection of *subject attributes* associated with the Web Application instance, the collection of *resource attributes* associated with the attempted operation, and the collection of *environment attributes* associated with the circumstances of the attempt. The determinedness of each of these attributes is in accordance with the *execution phase* of the attempt.

A *query* is evaluated against a *policy-set*, resulting in a *decision* in accordance with the evaluation rules defined in this specification.

Appendix C. SECURITY POLICY DOCUMENT FORMAT

This section is normative.

C.1 INTRODUCTION

This appendix defines a method for representing a Security Policy (e.g. for interchange or device management purposes).

C.2 SCHEMA

C.2.1 <SIGNED-POLICY>

The root element of a *signed policy document* is a `<signed-policy>`.

`<signed-policy>` contains, in any order, exactly one `<signature>` element and one or more elements each of which is either `<policy-set>` or `<policy>`.

C.2.2 <SIGNATURE>

The `<signature>` element, as a child of `<signed-policy>`, specifies the detached digital signature of the signed policy document as defined in XML Digital Signature⁸, with the following additional constraints:

- the `<signature>` element **MUST** contain one or more valid `<Reference>` elements;
- the URL attribute of each `<Reference>` element **MUST** contain a reference to a `<policy>` or `<policy-set>` element that is a sibling of the `<signature>` element in the same Signed Policy Document;
- the `<Reference>` element **MUST NOT** have any `<Transform>` elements;
- the Widget User Agent **MUST** treat the `<signed-policy>` as invalid if it has a child `<policy>` or `<policy-set>` element for which there is no `<Reference>` element.

Processing of the signature is specified in section 6.2.1.

C.2.3 <POLICY-SET>

The root element of a *policy document* is either a `<policy-set>` or a `<policy>`.

⁸ <http://www.w3.org/TR/xmlsig-core/>

<policy-set> has two possible attributes:

- combine, which must take a value of "deny-overrides", "permit-overrides" or "first-matching-target". The attribute is optional; if it is omitted, the default value is "deny-overrides";
- id, whose value is a textual identifier for the <policy-set>.

<policy-set> contains an optional <target>, then zero or more <policy> and/or <policy-set> elements.

C.2.4 <POLICY>

The root element of a *policy document* is either a <policy-set> or a <policy>.

<policy> has three possible attributes:

- combine, which must take a value of "deny-overrides", "permit-overrides" or "first-applicable". The attribute is optional; if it is omitted, the default value is "deny-overrides";
- description, whose value is a textual description of the policy;
- id, whose value is a textual identifier for the <policy-set>.

<policy> contains an optional <target>, then zero or more <rule> elements.

C.2.5 <RULE>

<rule> has one possible attribute, effect, which must take a value of "permit", "prompt-blanket", "prompt-session", "prompt-oneshot" or "deny". The attribute is optional; if it is omitted, the default value is "permit".

<rule> contains an optional <condition>.

C.2.6 <TARGET>

<target> contains one or more <subject> elements.

C.2.7 <SUBJECT>

<subject> contains one or more <subject-match> elements.

C.2.8 <CONDITION>

<condition> has one possible attribute, combine, which must take a value of "and" or "or". The attribute is optional; if it is omitted, the default value is "and".

<condition> contains one or more elements, each of which is one of <condition>, <subject-match>, <resource-match> or <environment-match>.

C.2.9 <SUBJECT-MATCH>, <RESOURCE-MATCH>, <ENVIRONMENT-MATCH>

<subject-match> represents a condition on a single subject attribute to be matched in a target or condition. <resource-match> represents a condition on a single resource attribute to be matched in a condition. <environment-match> represents a condition on a single environment attribute to be matched in a condition.

The element has up to three (XML) attributes:

- (mandatory) attr: the name of the subject, resource or environment (respectively) attribute to check. The attribute name is optionally followed by one of the suffixes ".scheme", ".authority", ".scheme-authority", ".host" or ".path", causing the equivalently named URI modifier function to be applied to the attribute value before matching.
- (optional) match: the literal text to match. If this attribute is omitted, then the contents of the element are used instead.
- (optional) func: the match function to use. If it is present, it must be one of "equal", "glob" or "regexp". If func is omitted, then the default is "glob".

The contents of a <subject-match>, <resource-match> or <environment-match> element represent the value to match, only if the match attribute is absent. If the match attribute is present, then the element contents are ignored. For a <subject-match>, the contents are PCDATA. For the others, the contents can be any combination of PCDATA, <subject-attr>, <resource-attr> and <environment-attr> elements, giving the literal text to match after expanding any attributes.

C.2.10 <SUBJECT-ATTR>, <RESOURCE-ATTR>, <ENVIRONMENT-ATTR>

Each of these elements represents the value of a subject, resource or environment attribute respectively.

The element has one (XML) attribute, attr, giving the name of the attribute to expand. It has no contents.

----- END OF DOCUMENT -----