



Symptoms Automation Framework (SAF) Version 1.0

Candidate OASIS Standard 02

04 September 2013

Specification URIs

This version:

<http://docs.oasis-open.org/saf/saf/v1.0/cos02/saf-v1.0-cos02.doc> (Authoritative)
<http://docs.oasis-open.org/saf/saf/v1.0/cos02/saf-v1.0-cos02.html>
<http://docs.oasis-open.org/saf/saf/v1.0/cos02/saf-v1.0-cos02.pdf>

Previous version:

<http://docs.oasis-open.org/saf/saf/v1.0/cs01/saf-v1.0-cs01.doc> (Authoritative)
<http://docs.oasis-open.org/saf/saf/v1.0/cs01/saf-v1.0-cs01.html>
<http://docs.oasis-open.org/saf/saf/v1.0/cs01/saf-v1.0-cs01.pdf>

Latest version:

<http://docs.oasis-open.org/saf/saf/v1.0/saf-v1.0.doc> (Authoritative)
<http://docs.oasis-open.org/saf/saf/v1.0/saf-v1.0.html>
<http://docs.oasis-open.org/saf/saf/v1.0/saf-v1.0.pdf>

Technical Committee:

OASIS Symptoms Automation Framework (SAF) TC

Chairs:

Jeffrey Vaught (Jeffrey.Vaught@ca.com), CA Technologies
Stavros Isaiadis (stavros.isaiadis@baml.com), Bank of America Merrill Lynch

Editor:

Vivian Lee (Vivian.Lee@uk.fujitsu.com), Fujitsu Limited

Additional artifacts:

This prose specification is one component of a Work Product which also includes:

- XML schemas: <http://docs.oasis-open.org/saf/saf/v1.0/cos02/schemas/>

Declared XML namespace:

- <http://docs.oasis-open.org/saf/ns/symptoms/2012/07>

Abstract:

This document normatively defines a reference architecture for the Symptoms Automation Framework, a tool in the automatic detection, optimization, and remediation of operational aspects of complex systems, notably data centers. It also provides a non-normative XML data model, based on a pseudo schema and an XSD.

Status:

This document was last revised or approved by the OASIS Symptoms Automation Framework (SAF) TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the

“Send A Comment” button on the Technical Committee’s web page at <http://www.oasis-open.org/committees/saf/>.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (<http://www.oasis-open.org/committees/saf/ipr.php>).

Citation format:

When referencing this specification the following citation format should be used:

[SAF-v1.0]

Symptoms Automation Framework (SAF) Version 1.0. 04 September 2013. Candidate OASIS Standard 02. <http://docs.oasis-open.org/saf/saf/v1.0/cos02/saf-v1.0-cos02.html>.

Notices

Copyright © OASIS Open 2013. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full [Policy](#) may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of [OASIS](#), the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <http://www.oasis-open.org/policies-guidelines/trademark> for above guidance.

Table of Contents

1	Introduction.....	5
1.1	Terminology.....	5
1.1.1	Notational Conventions.....	5
1.2	XML Namespaces.....	5
1.3	Normative References.....	6
2	Information Model.....	7
2.1	SAFType.....	7
2.1.1	Non Normative Pseudo Schema.....	8
2.2	Syndrome.....	8
2.2.1	Non Normative Pseudo Schema.....	9
2.3	Protocol.....	10
2.3.1	Non Normative Pseudo Schema.....	12
2.4	ProtocolGroup.....	14
2.4.1	Non Normative Pseudo Schema.....	14
2.5	Prescription.....	15
2.5.1	Non Normative Pseudo Schema.....	16
2.6	Symptom.....	17
2.6.1	Non Normative Pseudo Schema.....	18
2.7	SymptomSchema.....	19
2.7.1	Non Normative Pseudo Schema.....	19
2.8	PrescriptionSchema.....	20
2.8.1	Non Normative Pseudo Schema.....	20
3	Architectural Roles.....	21
3.1	Information Sources.....	21
3.1.1	Syndrome and Protocol Catalog.....	21
3.1.2	Symptom Store.....	21
3.2	Active Roles.....	21
3.2.1	Catalog Source.....	21
3.2.2	Symptom Source.....	22
3.2.3	Case Manager.....	22
3.2.4	Diagnostician.....	22
3.2.5	Practitioner.....	23
4	Interfaces.....	24
5	Notes on Future Specification Development.....	26
6	Examples.....	27
6.1	Medical Sequence Diagram.....	27
6.2	Catalogue Authoring Diagram.....	28
7	Conformance.....	30
Appendix A.	Acknowledgements.....	31
Appendix B.	Revision History.....	32

1 Introduction

The Symptoms Automation Framework is architecture for enabling interoperable diagnosis and treatment of complex systems. The architecture is implementation agnostic yet it supports both stateful or real-time processing and postmortem diagnostics. The key constituent of the architecture is the Symptom, an instance indicating an observed negative or positive condition. Symptoms can be characterized by a Syndrome, which is a published pattern of conditions and other Symptoms. Once identified, a Syndrome can be treated (either to remedy a problem or enhance positive characteristics of the system) by application of one or more Protocols, which describe how to carry out a process to treat, optimize, or further diagnose the Syndrome. The Protocol is rendered to a specific situation and subject in the form of a Prescription. The framework also provides for diagnostics, a type of Protocol, to provide further information to refine the diagnosis of a given Syndrome. These four main elements comprise the Symptoms information model, presented in the next section. This document also defines the key actors that participate in the Symptoms cycle of identify, diagnose, and treat, namely the Syndrome Catalog, Case Manager, Symptom Source, Diagnostician, and Practitioner. Lastly, a collection of interfaces, which may be supported by these actors, is described.

1.1 Terminology

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 [RFC2119].

1.1.1 Notational Conventions

This specification uses a notational convention, referred to as “Pseudo-schemas” in a fashion similar to the WSDL 2.0 Part 1 specification. A Pseudo-schema uses a BNF-style convention to describe attributes and elements:

- ‘?’ denotes optionality (i.e. zero or one occurrences),
- ‘*’ denotes zero or more occurrences,
- ‘+’ one or more occurrences,
- ‘[’ and ‘]’ are used to form groups,
- ‘|’ represents choice.
- Attributes are conventionally assigned a value corresponding to their type.

```
<!-- sample pseudo-schema -->
<element
  required attribute of type QName="xs:QName"
  optional attribute of type string="xs:string"? >
<required element />
<optional element />?
<one or more of these elements />+
<zero or more of these elements />*
  [ <choice 1 /> | <choice 2 /> ]
</element>
```

1.2 XML Namespaces

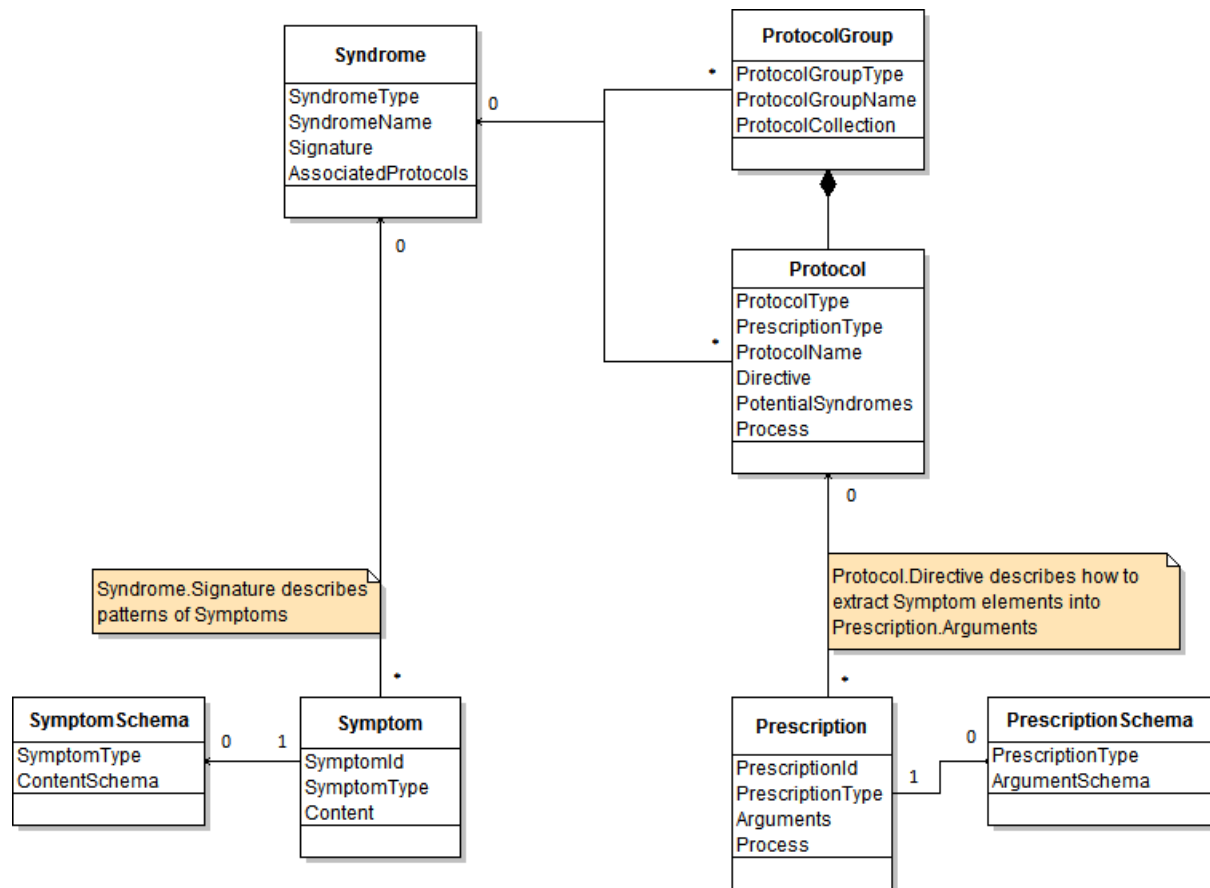
The following namespaces are used in this document:

Prefix	Namespace
xsd	http://www.w3.org/2001/XMLSchema
saf	http://docs.oasis-open.org/saf/ns/symptoms/2012/07

43 **1.3 Normative References**

44 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
45 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
46
47 **[XPath20]** A. Berglund et al, XML Path Language, version 2.0,
48 <http://www.w3.org/TR/xpath20/>, January 2007.
49
50 **[XQuery]** S. Boag et al, XQuery 1.0: An XML Query Language,
51 <http://www.w3.org/TR/xquery/>, January 2007.
52
53 **[XML10]** T. Bray et al, Extensible Markup Language (XML) 1.0, November 2008.
54
55

2 Information Model



57

2.1 SAFType

58 The SAFType is a definition used throughout the specification to represent a unique semantics for an
 59 element.
 60

Field	Type	Properties	Description
Uri	anyURI	Required, Immutable	The Uri, uniquely defines the semantics of the SAFType.
Version	string	Optional, Immutable	The Version, in combination with Uri to establish supplemental uniqueness of a SAFType.

61

62 2.1.1 Non Normative Pseudo Schema

63 The following is one possible non-normative pseudo schema for the SAFType.

```
64 <SAFType>  
65 <Uri>xsd:anyURI</Uri>  
66 <Version>xsd:string</Version>?  
67 </SAFType>
```

68
69 Example of a SAFType for a Fever Syndrome:

```
70 <SAFType>  
71 <Uri>http://example.com/saf/types/syndromes/fever/</Uri>  
72 <Version>1</Version>  
73 </SAFType>
```

74

75 2.2 Syndrome

76 A Syndrome is an identifiable collection of zero or more related Symptoms (as identified by a signature).
77 Since a Syndrome describes a Symptom (see below) a Syndrome can be thought of as describing a class
78 of Symptom Instances.

Field	Type	Properties	Description
Syndrome Type	SAFType	Required, Immutable	The SyndromeType uniquely defines the semantics of the Syndrome.
Syndrome Name	string	Required, Mutable	A descriptive name for the Syndrome.
Description	string	Required, Mutable	A verbose explanation of the Syndrome for human consumption.
Likelihood	{VeryFrequent, Frequent, Balanced, Infrequent, Rare, NotAvailable}	Required, Mutable	An indication as to the typicality of this Syndrome.
Impact	{VeryHigh, High, Moderate, Low, Minimal, Unknown}	Required, Mutable	The effect of this Syndrome with respect to the consequences of not detecting, diagnosing, or treating it.
Urgency	{VeryHigh, High, Moderate, Low, VeryLow, Unknown}	Required, Mutable	The speed and tenacity with which this Syndrome should receive attention.
Signature	string	Required, Mutable	An XQuery expression [XQUERY] that detects an interesting pattern of Symptoms and defines how to recognize a Syndrome. If the result is empty the Syndrome is not present in the system. A non-empty result contains a valid XML document [XML10]. This document MAY contain matched Symptom instances or other information pertaining to the Syndrome. This document MUST be available for Processing Protocols.
Associated Protocols	ProtocolReference [0..n]	Optional, Mutable	A collection of diagnostic tests and actions, of which NONE or ONE may be prescribed by the Diagnostician. The list may contain zero or

	ProtocolGroup Reference[0..n]		more Protocols and/or zero or more groups of Protocols. Protocols within a group are executed together.
--	-------------------------------	--	---

79 2.2.1 Non Normative Pseudo Schema

80 The following is one possible non-normative pseudo schema for the Syndrome.

```

81 <Syndrome>
82   <SyndromeType>saf:SAFType</SyndromeType>
83   <SyndromeName>xsd:string</SyndromeName>
84   <Description>xsd:string</Description>
85   <Likelihood>
86     [Common|Uncommon|Rare|NotAvailable]
87   </Likelihood>
88   <Impact>
89     [HighImpact|ModerateImpact|LowImpact|UnknownImpact]
90   </Impact>
91   <Urgency>
92     [HighUrgency|ModerateUrgency|LowUrgency|UnknownUrgency]
93   </Urgency>
94   <Signature>xsd:string</Signature>
95   <AssociatedProtocols>
96     <ProtocolReference>saf:SAFType</ProtocolReference>*
97     <ProtocolGroupReference>saf:SAFType</ProtocolGroupReference>*
98   </AssociatedProtocols>?
99 </Syndrome>

```

100

101 Example of a Syndrome to identify Fever based on a temperature values (Symptoms) coming from
102 sensors. The associated protocols will attempt a remediation, perhaps without fully understanding the
103 symptoms, by giving aspirin, and also perform more diagnostic tests via the protocol group to determine
104 the cause of the fever.

```

105 <Syndrome>
106   <SyndromeType>
107     <Uri>http://example.com/saf/types/syndromes/fever/</Uri>
108     <Version>2</Version>
109   </SyndromeType>
110   <SyndromeName>FeverSyndrome</SyndromeName>
111   <Description>Syndrome identifying fever</Description>
112   <Likelihood>Common</Likelihood>
113   <Impact>Low</Impact>
114   <Urgency>Moderate</Urgency>
115   <Signature>
116     for $x in /Symptoms/Symptom
117     where
118       $x[SymptomType="http://example.com/saf/types/symptoms/temperature/"
119         and $x/Content/Temperature[Value >= 38]
120     return $x
121   </Signature>
122   <AssociatedProtocols>
123     <ProtocolReference>
124       <Uri>http://example.com/saf/types/protocols/aspirin/</Uri>
125     </ProtocolReference>
126     <ProtocolGroupReference>
127       <Uri>
128         http://example.com/saf/types/protocol-groups/diagnosefever/
129       </Uri>
130     </ProtocolGroupReference>
131   </AssociatedProtocols>
132 </Syndrome>
133

```

134 **2.3 Protocol**

135 A Protocol is a process for confirming a potential Syndrome diagnosis via the creation of validating
 136 Symptoms, for remediating a Syndrome, optimizing the system, and/or preventing a Syndrome from
 137 occurring. It provides the template necessary to create a Prescription.

Field	Type	Properties	Description
Protocol Type	SAFType	Required, Immutable	ProtocolType uniquely defines the semantics of the Protocol.
Prescription Type	SAFType	Required, Immutable	PrescriptionType uniquely defines the semantics of all Prescription instances, baring this type, created as a result of applying this Protocol and MUST be included in any generated Prescriptions.
Protocol Name	string	Required, Mutable	A descriptive name for the Protocol.
Description	string	Required, Mutable	A verbose explanation of the Protocol for human consumption.
Effectiveness	{Effective, PartiallyEffective, BestEffort, Ineffective, Inconclusive, Unknown}	Required, Mutable	The expected success of the Protocol.
Risk	{VeryHigh, High, Moderate, Low, VeryLow, Unknown}	Required, Mutable	The expected side effects or undesired consequences of running the Protocol.
Duration	{VeryLong, Long, Moderate, Short, VeryShort, Unknown}	Required, Mutable	The expected amount of time necessary to complete the Protocol.
Function	{Diagnostic, Preventative, Remedial, Diagnostic_Preventative, Diagnostic_Remedial, Preventative_Remedial, Diagnostic_Preventative_Remedial, Other}	Required, Mutable	The type of Protocol, either a remedial treatment, a preventative treatment, a confirming diagnostic, or a combination.
Directive	string	Required, Mutable	An XQUERY expression that generates an XML document containing information needed to create the Arguments of a Prescription instance. This document MAY contain Symptom elements or other information

			pertaining to the Syndrome or the system environment.
Potential Syndromes	Syndrome Reference[0..n]	Optional, Mutable	A list of Syndromes that can be indirectly matched as a result of the Protocol process.
Process	string	Optional, Mutable	Implementation specific diagnostic and treatment workflow instructions.

138 2.3.1 Non Normative Pseudo Schema

139 The following is one possible non-normative pseudo schema for the Protocol class.

```
140 <Protocol>
141   <ProtocolType>saf:Type</ProtocolType>
142   <PrescriptionType>saf:Type</PrescriptionType>
143   <ProtocolName>xsd:string</ProtocolName>
144   <Description>xsd:string</Description>
145   <Effectiveness>
146     [Effective|PartiallyEffective|
147      BestEffort|Ineffective|Inconclusive]
148   </Effectiveness>
149   <Risk>
150     [HighRisk|ModerateRisk|LowRisk|UnknownRisk]
151   </Risk>
152   <Duration>
153     [LongDuration|ModerateDuration|ShortDuration|UnknownDuration]
154   </Duration>
155   <Function>
156     [Diagnostic|Preventative|Remedial|
157      Diagnostic Preventative|Diagnostic Remedial|
158      Preventative Remedial|Diagnostic Preventative Remedial|Other]
159   </Function>
160   <Directive>xsd:string</Directive>
161   <PotentialSyndromes>
162     <SyndromeReference>saf:SAFType</SyndromeReference>*
163   </PotentialSyndromes>?
164   <Process>xsd:string</Process>?
165 </Protocol>
```

166

167 Example of a Protocol designed to provide temporary remediation of the Fever Syndrome.

```
168 <Protocol>
169   <ProtocolType>
170     <Uri>http://example.com/saf/types/protocols/aspirin/</Uri>
171   </ProtocolType>
172   <PrescriptionType>
173     <Uri>
174       http://example.com/saf/types/prescriptions/aspirin/
175     </Uri>
176   </PrescriptionType>
177   <ProtocolName>AspirinProtocol</ProtocolName>
178   <Description>Medication for Fever</Description>
179   <Effectiveness>BestEffort</Effectiveness>
180   <Risk>Low</Risk>
181   <Duration>Short</Duration>
182   <Function>Remedial</Function>
183   <Directive>
184     for $x in /Symptoms/Symptom
185       let $subject := $x/Subject
186       let $value := fn:number($x/Content/Temperature/Value)
187       return
188         <Details>
189           <Subject>$subject</Subject>
190           (: Give 1 aspirins for every 2 degrees above 38 :)
191           <AspirinCount>
192             {if ($value > 38) then (
193               fn:floor($value - 38) div 2)
194              else (0)}
195           </AspirinCount>
196         </Details>
197   </Directive>
198   <PotentialSyndromes>
199     <SyndromeReference>
200     <Uri>
```

```
201     http://example.com/saf/types/syndromes/fever/
202     </Uri>
203     </SyndromeReference>
204 </PotentialSyndromes>
205 <Process>
206     ProvisionAspirin (Subject, AspirinCount);
207 </Process>
208 </Protocol>
```

209 2.4 ProtocolGroup

210 A ProtocolGroup is a collection of Protocols which will be enacted together as a group. The Syndrome
211 AssociatedProtocols field references Protocol and/or ProtocolGroup allowing for flexibility in how
212 validation, remediation, optimization, and prevention processes are invoked.

Field	Type	Properties	Description
Protocol Group Type	SAFType	Required, Immutable	ProtocolGroupType uniquely defines the semantics of the ProtocolGroup.
Protocol Group Name	string	Required, Mutable	A descriptive name for the ProtocolGroup.
Description	string	Required, Mutable	A verbose explanation of the ProtocolGroup for human consumption.
Protocol Collection	ProtocolReference [1..n]	Required, Mutable	A collection of one or more Protocols which must be enacted together as a group.

213 2.4.1 Non Normative Pseudo Schema

214 The following is one possible non-normative pseudo schema for the ProtocolGroup class.

```
215 <ProtocolGroup>  
216   <ProtocolGroupType>saf:SAFType</ProtocolGroupType>  
217   <ProtocolGroupName>xsd:string</ProtocolGroupName>  
218   <Description>xsd:string</Description>  
219   <ProtocolCollection>  
220     <ProtocolReference>saf:SAFType</ProtocolReference>+  
221   </ProtocolCollection>  
222 </ProtocolGroup>
```

223

224 Example of a Protocol designed to gather more information in the fever diagnosis.

```
225 <ProtocolGroup>  
226   <ProtocolGroupType>  
227     <Uri>http://example.com/saf/types/protocol-groups/diagnosefever/</Uri>  
228   </ProtocolGroupType>  
229   <ProtocolGroupName>FeverDiagnosis</ProtocolGroupName>  
230   <Description>  
231     Actions necessary to diagnose the type of fever.  
232   </Description>  
233   <ProtocolCollection>  
234     <ProtocolReference>  
235       <Uri>http://example.com/saf/types/protocols/blood culture/</Uri>  
236     </ProtocolReference>  
237     <ProtocolReference>  
238       <Uri>http://example.com/saf/types/protocols/skin temperature/</Uri>  
239     </ProtocolReference>  
240   </ProtocolCollection>  
241 </ProtocolGroup>
```

242

243 **2.5 Prescription**

244 A Prescription is an instance of a process, which MAY correspond to a Protocol. It is used to provide
 245 remediation, diagnostics, preventative measures, or optimization to be performed. Prescriptions MAY
 246 represent automated or Manual steps. A Prescription includes arguments specific to the subject or
 247 component that is the target of the prescription.

Field	Type	Properties	Description
PrescriptionId	anyURI	Required, Unique, Immutable	The identifier for the Prescription. This element MUST be globally unique and MAY be used as the primary key for the Prescription.
Prescription Type	SAFType	Required, Immutable	The PrescriptionType defines the semantics of this Prescription. All Prescriptions bearing the same PrescriptionType MUST have the same semantics.
Expiration Date	dateTime	Optional, Mutable	An optional date recommendation beyond which the Prescription MAY no longer be useful.
Arguments	any	Optional, Mutable	The XML rendered arguments needed by the recipient of the Prescription to apply this Prescription to a specific target.
Process	string	Optional, Mutable	Optional process, such as a script to be executed by the recipient of the Prescription.

248 2.5.1 Non Normative Pseudo Schema

249 The following is one possible non-normative pseudo schema for the Prescription class.

```
250 <Prescription>
251   <PrescriptionId>xsd:anyURI</PrescriptionId>
252   <PrescriptionType>saf:SAFType</PrescriptionType>
253   <ExpirationDate>
254     xsd:dateTime
255   </ExpirationDate>?
256   <Arguments>xsd:any</Arguments>?
257   <Process>xsd:string</Process>?
258 </Prescription>
```

259

260 Example of a generated prescription that would check the arguments supplied and take the necessary
261 (simplistic in this case) decisions.

```
262 <Prescription>
263   <PrescriptionId>
264     http://example.com/saf/prescriptions/aspirin/12345
265   </PrescriptionId>
266   <PrescriptionType>
267     <Uri>http://example.com/saf/types/prescriptions/aspirin/</Uri>
268     <Version>2</Version>
269   </PrescriptionType>
270   <ExpirationDate>2011-10-23</ExpirationDate>
271   <Arguments>
272     <Details>
273       <Subject>http://example.com/saf/subjects/patient-234</Subject>
274       <AspirinCount>2</AspirinCount>
275     </Details>
276   </Arguments>
277   <Process>
278     ProvisionAspirin(Subject, AspirinCount);
279   </Process>
280 </Prescription>
281
```


282 **2.6 Symptom**

283 A Symptom is the instance, possibly corresponding to a Syndrome and described by a Signature,
 284 indicating that the condition or situation is present in the system. There SHOULD be a Syndrome
 285 corresponding to each type of Symptom or a combination of Symptoms as identified by the Syndrome
 286 signature. Unlike Syndromes and Protocols, which may be relatively static and represent the knowledge
 287 within the framework, Symptoms represent the dynamic state of the system and are therefore expected to
 288 be emitted frequently. Once emitted, Symptoms are immutable, and they can be safely used for audit
 289 trails and historical record keeping.

290 Symtoms may be linked to other previously emitted symptoms by specifying the unique ID of those
 291 symptoms and the type of relationship to them (e.g. causal, supersedes, custom, etc). Symptoms may
 292 also be associated with other symptoms in a less direct manner through one or more incident IDs.

Field	Type	Properties	Description
SymptomId	anyURI	Required, Unique, Immutable	The identifier for the Symptom. This element MUST be globally unique and MAY be used as the primary key for the Symptom.
Symptom Type	SAFType	Required, Immutable	This SymptomType defines the semantics of this Symptom. All Symptoms baring the same SymptomType MUST have the same semantics.
CreationDate	dateTime	Required, Immutable	The date-time (in UTC) when the Symptom was created. The value of this element SHOULD provide as much granularity as possible.
Confidence	{HighConfidence, ModerateConfidence, LowConfidence, UnknownConfidence}	Require, Immutable	The level of confidence in the accuracy and quality of this symptom, as determined by the Symptom Source.
Reporter	anyURI	Required, Immutable	Identification of the entity that emitted the Symptom.
Subject	anyURI	Required, Immutable	Identification of the entity exhibiting the Symptom.
Expiration Date	dateTime	Optional, Immutable	An optional date-time (in UTC) recommendation beyond which the Symptom may no longer be useful.
Related Symptoms	RelatedSymptom [0..n]	Optional, Immutable	A collection of previously emitted symptoms that are related to this symptom in one of a number of possible relationship types. The Symptom Emitter supplies this information.
Incident	anyURI [0..n]	Optional, Immutable	A Symptom Emitter can fill in this information denoting this Symptom to be part of a group of Symptoms all of which relate to the same incident.
Content	any	Optional, Immutable	An implementation dependent element that could contain such data as the raw events/messages that triggered the creation of the Symptom.

293 2.6.1 Non Normative Pseudo Schema

294 The following is one possible non-normative pseudo schema for the Symptom class.

```
295 <Symptom>
296   <SymptomId>xsd:anyURI</SymptomId>
297   <SymptomType>saf:SAFType</SymptomType>
298   <CreationDate>xsd:dateTime</CreationDate>
299   <Confidence>
300     [HighConfidence|ModerateConfidence|
301     LowConfidence|UnknownConfidence]
302   </Confidence>
303   <Reporter>xsd:anyURI</Reporter>
304   <Subject>xsd:anyURI</Subject>
305   <ExpirationDate>
306     xsd:DateTime
307   </ExpirationDate>?
308   <RelatedSymptoms>
309     <RelatedSymptom type="[Causal|Supersedes|Repetition|Other]">
310       xsd:anyURI
311     </RelatedSymptom>+
312   </RelatedSymptoms>?
313   <Incident>xsd:anyURI</Incident>?
314   <Content>xsd:any</Content>?
315 </Symptom>
```

316

317 Example of a symptom instance conveying temperature information from a sensor.

```
318 <Symptom>
319   <SymptomId>
320     http://example.com/saf/symptoms/temperature/2
321   </SymptomId>
322   <SymptomType>
323     <Uri>http://example.com/saf/types/symptom/temperature/</Uri>
324   </SymptomType>
325   <CreationDate>2011-10-24 13:10:05</CreationDate>
326   <Confidence>High</Confidence>
327   <Reporter>http://example.com/saf/reporters/tempsensor-123/</Reporter>
328   <Subject>http://example.com/saf/subjects/patient-234/</Subject>
329   <ExpirationDate>2011-10-24 14:10:05</ExpirationDate>
330   <RelatedSymptoms>
331     <RelatedSymptom type="Supersedes">
332       http://example.com/saf/symptoms/temperature/1
333     </RelatedSymptom>
334   </RelatedSymptoms>
335   <Incident>http://example.com/saf/incidents/12345</Incident>
336   <Content>
337     <Temperature>
338       <Value>38</Value>
339       <Scale>C</Scale>
340     </Temperature>
341   </Content>
342 </Symptom>
```

343

344

345

346

347 2.7 SymptomSchema

348 A SymptomSchema describes the non-normative xml in the Content field of Symptoms. With this
349 information, a catalog author has the complete picture of a Symptom definition for a given type, and is
350 able to create Syndrome signatures describing patterns of interest within a collection of Symptoms.

351 The SymptomSchema entity is entirely optional within a SAF system, as the information needed to create
352 Syndrome signatures could be gleaned from existing Symptoms in the SymptomStore. The
353 SymptomSchema offers a more straightforward way of defining that information. One that doesn't require
354 the pre-existence of Symptoms.

355 SymptomSchema is most closely aligned with the role of Symptom Source. These sources can optionally
356 register SymptomSchema entries into the Catalog for each type of Symptom.

Field	Type	Properties	Description
Symptom Type	SAFType	Required, Unique, Immutable	This SymptomType defines the semantics of this SymptomSchema. All SymptomSchemas baring the same SymptomType MUST have the same semantics.
Content Schema	any	Required	Describes the Symptom Content xml for this type via XML Schema Document notation.

357 2.7.1 Non Normative Pseudo Schema

358 The following is one possible non-normative pseudo schema for the SymptomSchema class.

```
359 <SymptomSchema>  
360   <SymptomType>saf:SAFType</SymptomType>  
361   <ContentSchema>xsd:any</ContentSchema>  
362 </SymptomSchema>
```

363

364 Example of a schema for temperature information.

```
365 <SymptomSchema>  
366   <SymptomType>  
367     <Uri>http://example.com/saf/types/symptom/temperature/</Uri>  
368   </SymptomType>  
369   <ContentSchema>  
370     <Temperature>  
371       <Value>xsd:float</Value>  
372       <Scale>[C|F]</Scale>  
373     </Temperature>  
374   </ContentSchema>  
375 </SymptomSchema>
```

376

377 2.8 PrescriptionSchema

378 A PrescriptionSchema describes the non-normative xml in the Arguments field of Prescriptions. With this
379 information, a catalog author has the complete picture of a Prescription definition for a given type, and is
380 able to create the Protocol directives used to translate pattern results into Prescription arguments.

381 The PrescriptionSchema entity is entirely optional within a SAF system, as the information needed to
382 create Protocol directives could be manually gleaned from external Practitioner documents and so forth.
383 The PrescriptionType offers a more straightforward way of defining that information.

384 PrescriptionSchema is most closely aligned with the role of Practitioner. The Practitioner can optionally
385 register PrescriptionSchema entries into the Catalog for each type of Prescription.

Field	Type	Properties	Description
Prescription Type	SAFType	Required, Unique, Immutable	The PrescriptionType defines the semantics of this PrescriptionSchema. All PrescriptionSchemas bearing the same PrescriptionType MUST have the same semantics.
Argument Schema	any	Required	Describes the Prescription Argument xml for this type via XML Schema Document notation.

386 2.8.1 Non Normative Pseudo Schema

387 The following is one possible non-normative pseudo schema for the Prescription class.

```
388 <PrescriptionSchema>  
389   <PrescriptionType>saf:SAFType</PrescriptionType>  
390   <ArgumentSchema>xsd:any</ArgumentSchema>  
391 </PrescriptionSchema>
```

392

393 Example of a schema for the aspirin disposing Prescription.

```
394 <PrescriptionSchema>  
395   <PrescriptionType>  
396     <Uri>http://example.com/saf/types/prescriptions/aspirin/</Uri>  
397     <Version>2</Version>  
398   </PrescriptionType>  
399   <ArgumentSchema>  
400     <Details>  
401       <Subject>xsd:anyURI</Subject>  
402       <AspirinCount>xsd:integer</AspirinCount>  
403     </Details>  
404   </ArgumentSchema>  
405 </PrescriptionSchema>
```

406

3 Architectural Roles

407 An implementation of the Symptoms Automation Framework MAY implement any of the roles detailed
408 below. If an implementation provides a capability described by any of the roles, it MUST implement that
409 capability as specified below. An implementation MAY incorporate all the roles into a single entity or MAY
410 define separate entities for collections of roles. More than one instance of any role MAY be present in an
411 implementation of the Symptoms Automation Framework.

3.1 Information Sources

412 This specification defines two information sources, the Syndrome and Protocol Catalogue (Catalogue for
413 short), and the Symptom Store. This specification does not prescribe the method for persisting the
414 information sources (e.g. data base, files store, memory image, etc.). This specification prescribes the
415 contents of the data exchange and recommends a set of schemas by which this data is communicated to
416 and from other roles and components of the Symptoms Automation Framework.
417

3.1.1 Syndrome and Protocol Catalog

418 The Catalog contains Syndromes and Protocols associated with the system for which that Catalog was
419 designed, as well as SymptomSchema and PrescriptionSchema which define the schemata for the
420 Symptom content and the Prescription arguments respectively. In any Symptoms Automation Framework
421 there MAY be several Catalogs, each possibly associated with a specialized aspect of the system. While
422 the Catalog is generally static during operation of the Symptoms Automation Framework, it MAY evolve
423 over time as new Syndromes and Protocols are identified. The data exchange to and from the Catalog
424 MUST comply with the Syndromes and Protocols as defined in this specification.
425

3.1.2 Symptom Store

426 The Symptoms Store is an optional repository when Symptom persistence is desired and contains
427 Symptoms that have been created by Symptom sources. In any Symptoms Automation Framework there
428 MAY be several Symptom Stores. The Symptom Store is dynamic and its contents are expected to
429 change continuously during the operation of the Symptoms Framework. The currency of the Symptom
430 Store is dependent on many factors such as Symptom Source emission rate, network latency, store
431 frequency, etc. The data flows to and from the Symptoms Store MUST carry Symptoms as defined in this
432 specification.
433

3.2 Active Roles

434 The Active Roles in the Symptoms Automation Framework include Catalog Sources, Symptoms Sources,
435 a Case Manager, a Diagnostician, and a Practitioner described in the following sections. Each role MAY
436 be instantiated in the Symptoms Automation Framework as a distinct component. The roles MAY also be
437 combined in arbitrary ways to create more complex components performing the functions of several or all
438 roles. There MAY be any number (including zero) of components in Symptoms Automation Framework
439 exhibiting each role.
440

3.2.1 Catalog Source

441 The Catalog Source role represents a source of Syndromes and Protocols. A Catalog MAY have initial
442 content or be empty when Symptoms Automation Framework is setup. A Catalog Source MAY provide
443 additional contents to or updates the Catalogs as the Symptoms Automation Framework evolves during
444 operation.
445

446 **3.2.2 Symptom Source**

447 The Symptom Source role represents an emitter of Symptoms. A Symptom Source MAY provide
448 Symptoms at any time. The symptom source MAY be a component that experiences the symptom (the
449 *subject* or affected component) or the *reporter* of a symptom that receives, filter, enrich, and forwards,
450 symptoms from other Symptom Sources.

451 **3.2.3 Case Manager**

452 The Case Manager acts as the orchestrator within the Symptoms Automation Framework. The Case
453 Manager gathers Symptoms, keeps track of what Symptoms are currently of importance within the
454 system, and directs the actions of the other roles. The Case Manager maintains the state of the
455 Symptoms Automation Framework and keeps track of the diagnose-prescribe cycle. A Case Manager
456 may have broader knowledge about the entire system disposition and consult with one or more
457 Diagnosticians to leverage specialties prior to prescribing a Prescription. The Case Manager role selects
458 which Prescriptions to administer based on Diagnoses provided by the Diagnosticians. These
459 Prescriptions MAY provide additional diagnostic information (that is a new Symptom) to narrow the scope
460 of possible Syndromes or perform treatments on the system.

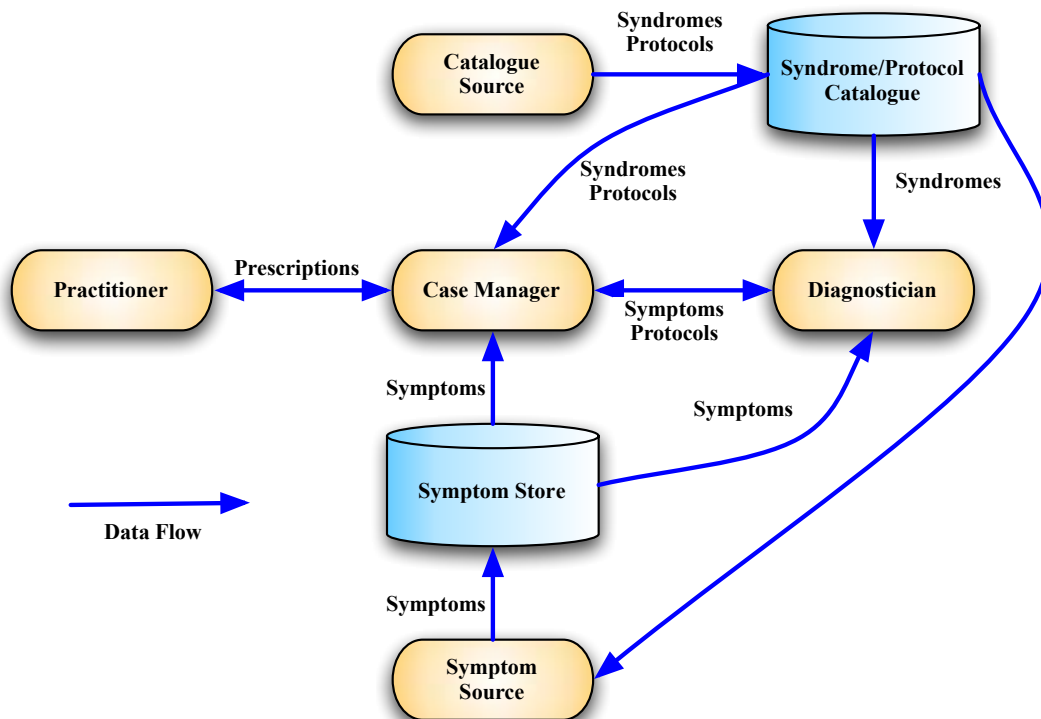
461 **3.2.4 Diagnostician**

462 The Diagnostician compares Symptoms with the signatures of various Syndromes to determine if any
463 Syndromes, matching those Symptoms, exist within the system. While the rules governing the processes
464 are expressed in XQuery, the processes used to analyze and/or match against the Syndromes are
465 implementation specific.

466

467 **3.2.5 Practitioner**

468 The Practitioner administers Prescriptions as requested by the Case Manager. There may be one or more
469 Practitioners in a SAF system, each one potentially able to understand and administer a different set of
470 PrescriptionTypes.



471

Figure 1: Roles and Information Stores in the Symptoms Automation Framework.

472 4 Interfaces

473 Problem determination includes problem detection, isolation, and resolution. Effective problem diagnosis
 474 is dependent upon basic reliability, availability, and serviceability (RAS) capabilities present in any
 475 system. Problems include situations that degrade the overall performance of installed components,
 476 situations that make some of the components unavailable, and situations that make all components
 477 unavailable. Often components implement special behavior that is available when they are in a failure
 478 mode. This behavior captures the internal and or external state of the component to aid in later problem
 479 determination.

480 The components can either play a role as the component that experiences the situation (the *subject* or
 481 affected component) or the *reporter* of a situation. In some cases, the reporter and the subject
 482 components can be the same. The subject and reporter roles are outside the Symptoms Automation
 483 Framework architecture, but are discussed here for clarity.

484 The Subject is the component that was affected by or was impacted by the event or the situation. The
 485 reporters are those components that submit symptoms on behalf of the Subjects. The reporter produces
 486 symptoms according to the symptoms model and uses an emission mechanism to submit the symptoms.

487 In this specification we have introduced concepts of the Symptom, Syndrome, Protocol, and Prescription
 488 each describing parts of the Symptoms Automation Framework information model. These elements of
 489 the information model are exchanged using the following interfaces.

490

Interface	Description	Candidate Role
Symptom Emitter	This is for the symptom sources or reporters emitting symptoms	Symptom Source
	Operations <ul style="list-style-type: none"> List supported types (Optional) 	

491

Symptom Handler	This is for the entity that receives symptoms for further processing	Diagnostician Symptom Source Case Manager Others
	<ul style="list-style-type: none"> Get a Symptom Add a Symptom Query Symptoms 	

492

Prescription Emitter	The source for emitting a prescription	Case Manager
	N/A	

493

Prescription Handler	This is for component that receives and acts on the prescription	Practitioner Case Manager Others
	<ul style="list-style-type: none"> Receive Prescription 	

	<ul style="list-style-type: none"> List supported types
--	--

494

Catalog Emitter	The source (files, tools, etc) for syndromes and protocols.	Catalog Source Authoring Tools
	N/A	

495

Catalog Handler	This for the component that is capable of handling specific syndromes and protocols.	Catalog Source Case Manager Others
	<ul style="list-style-type: none"> Get a Syndrome Add a Syndrome Update a Syndrome Delete a Syndrome Query Syndromes Get a Protocol Add a Protocol Update a Protocol Delete a Protocol Query Protocols Associate a Protocol to a Syndrome (Optional) Get a SymptomType Add a SymptomType Update a SymptomType Delete a SymptomType Get a PrescriptionType Add a PrescriptionType Update a PrescriptionType Delete a PrescriptionType 	

496 5 Notes on Future Specification Development

497 This section highlights a number of issues that the authors believe should be addressed by the Technical
498 Committee once it is formed. The reasons for not addressing these issues in this version of the
499 specification vary from, a feeling that a wider community is needed to address them, to a need to
500 complete this version in a timely manner.

501 While the Signature in a Syndrome is specified as a single XQuery expression, it is acknowledged by the
502 authors that processing of this expression may be performed incrementally to reflect the dynamic nature
503 of Symptom creation. It may be necessary to decompose, explicitly in the specification, this XQuery
504 expression into a conjunction of multiple, simpler expressions.

505 The Associated Protocols in a Syndrome may have dependencies between them, such as "all must be
506 applied", "any one may be applied", "must be applied in order", or possibly organized into sub-groups.

507 The current ProtocolGroup concept will handle the majority of cases where this is needed, but any more
508 sophisticated requirements will have to be defined more explicitly perhaps in a combination Protocol.

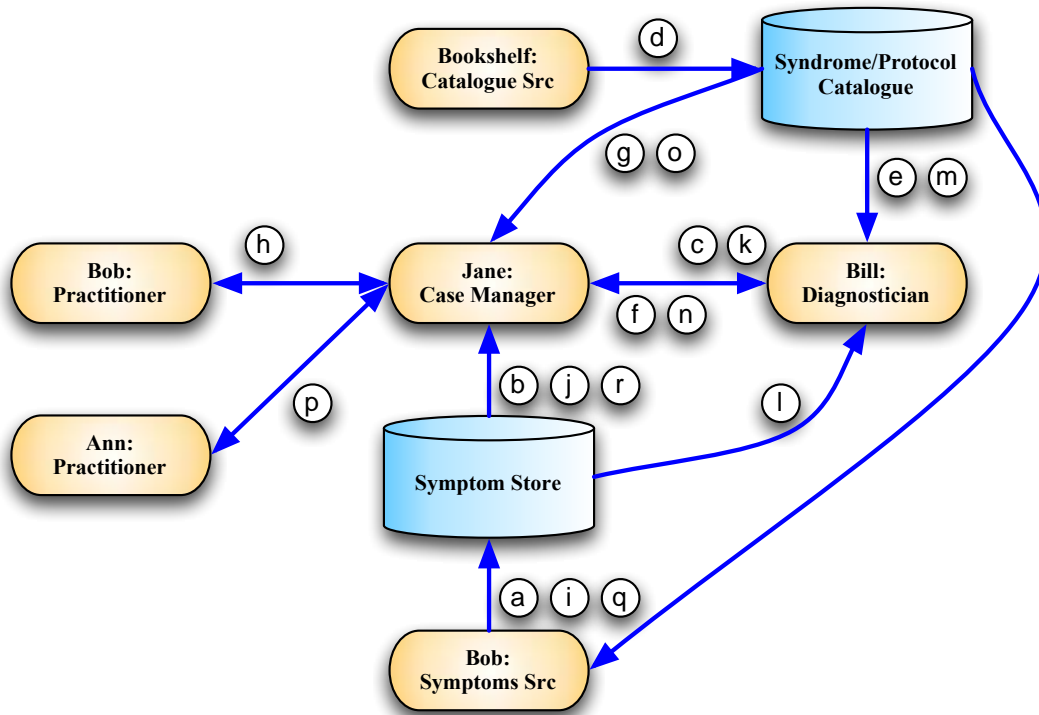
509 Extensibility in the specification is handled with the concepts of SymptomSchema and
510 PrescriptionSchema, which enable the modification of open content schemata to support custom
511 application requirements. In addition, the related symptoms type, which defines relationships between
512 symptoms, is also extensible in that it recommends a number of standard relations ("causal",
513 "supersedes", "repetition", etc) but allows any arbitrary values to be used. However, the above
514 notwithstanding, this specification could benefit even more from extensibility. Extensibility can help with
515 the development of future versions of the specification and possible extensions.

516

517 6 Examples

518 6.1 Medical Sequence Diagram

519 The diagram below provides non-normative example of how the Symptoms Automation Framework might
520 apply in the motivational use case used to design the Symptoms concept. This example is drawn from the
521 simple case of someone not feeling well and a health care provider provides diagnosis and treatment.



522

Figure 2: Medical Diagnosis Sequence

523 Symptoms Process:

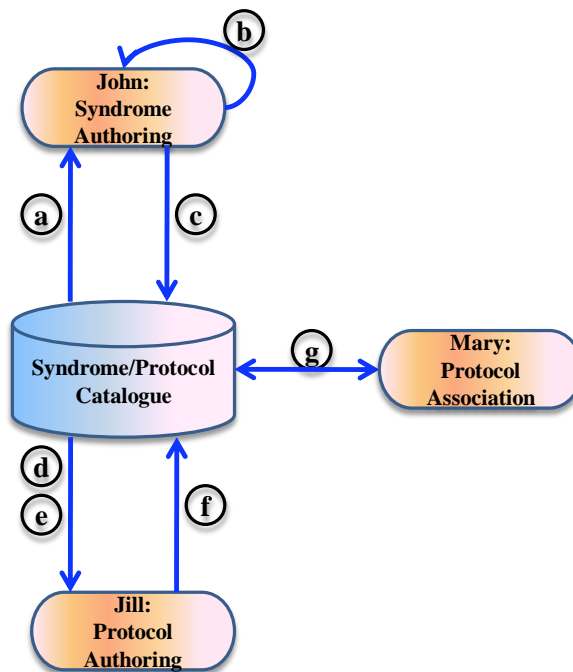
- 524 a) Bob (Symptoms Source) says, "I don't feel so good." (Symptom).
- 525 b) Jane (Case Manager) hears of Bob's Symptom, and
- 526 c) asks Bill (Diagnostician), "What do you think it is? "
- 527 d) Bill collects a first aid book (Catalog) from the bookshelf (Catalog Source).
- 528 e) Bill consults the Catalog and learns that the top entry listing the "I don't feel so good" Symptom
- 529 is a "Fever" (Syndrome), and
- 530 f) he passes this to Jane.
- 531 g) Jane looks up "Fever" in the Catalog where it recommends, "take temperature" (Protocol) to
- 532 verify the Syndrome, using a mercury thermometer (Prescription).
- 533 h) She then instructs Bob (this time a Practitioner) to take his own temperature.
- 534 i) Bob reports his temperature (a new Symptom).
- 535 j) Jane reads it and
- 536 k) again consults Bill.
- 537 l) Bill reads the value of the temperature and

- 538 m) again finds that "Fever" is the most likely Syndrome based on the high value of the newly
- 539 reported Symptom.
- 540 n) Bill tells Jane it's a "Fever."
- 541 o) Jane, again consulting the Catalog, decides that a medication (Protocol) is needed and selects
- 542 two Aspirin (Prescription) and
- 543 p) asks Ann to give Bob two Aspirin.
- 544 q) Bob later reports, "I feel much better" (another new Symptom) and Jane stops worrying.

545 6.2 Catalogue Authoring Diagram

546 The diagram below provides non-normative example of how Catalogue Authors may go about retrieving
 547 available Symptom and Prescription types in order to define Syndromes and Protocols.

548



549

550 *Figure 3: Catalog Authoring*

551

552

553 Authoring Process:

- 554 a) John (Catalogue Source) wants to define a syndrome for Fever. He consults the Catalogue
- 555 and finds the SymptomSchema used by Symptoms conveying temperature information
- 556 (added by Symptom Emitters able to emit temperature data)
- 557 b) He uses the schema to construct a signature for the Fever Syndrome
- 558 c) John publishes the Syndrome in the Catalogue
- 559 d) Jill (Catalogue Source) is responsible for defining appropriate Protocols and wants to define
- 560 one to tackle Fever. She searches for what type and format of arguments are expected in

561 order to generate a Prescription to remediate Fever. She finds a relevant PrescriptionSchema
562 in the Catalogue (as generated and added to the Catalogue by Practitioners that can handle
563 such Prescriptions).

- 564 e) Jill also needs to know how to extract these arguments, so she looks into the Fever
565 Syndrome's Signature to find out what it will return as a result.
- 566 f) Jill then creates a Protocol with a Directive able to generate the above PrescriptionSchema
567 by extracting Subject and AspirinCount information from the Symptoms returned by the
568 Syndrome signature. She adds this Protocol to the Catalogue.
- 569 g) Jill then goes on to associate this Protocol to the Fever Syndrome.

570 **7 Conformance**

571 An implementation is not conformant with this specification if it fails to satisfy one or more of the MUST or
572 REQUIRED level requirements defined herein for the roles and modes it implements.

573 Definitions in any associated XML Schemas are definitive and take precedence over conflicting definitions
574 in the main specification.

575

576 **Appendix A. Acknowledgements**

577 The following individuals have participated in the creation of this specification and are gratefully
578 acknowledged:

579 **Participants:**

580 Mike Baskey, IBM
581 Alvin Black, CA
582 Stavros Isaiadis, Bank of America Merrill Lynch (previously Fujitsu Limited)
583 Vivian Lee, Fujitsu Limited
584 Paul Lipton, CA
585 Yasuhide Matsumoto, Fujitsu Limited
586 Marcelo Perazolo, IBM
587 David Snelling, Fujitsu Limited
588 Jeffrey Vaught, CA

589 **Co-Developers of the initial contributions:**

590 This document is based on initial contributions to the OASIS SAF Technical Committee by the following
591 co-developers.

592 Mike Baskey, IBM
593 Alvin Black, CA
594 Vivian Lee, Fujitsu Limited
595 Paul Lipton, CA
596 Yasuhide Matsumoto, Fujitsu Limited
597 Marcelo Perazolo, IBM
598 Abdi Salahshour, IBM
599 David Snelling, Fujitsu Limited
600 Jeffrey Vaught, CA

601 **Acknowledgements of the initial contributions:**

602 The following individuals have provided invaluable input to the original contributions and were
603 acknowledged in the initial contributions.

604 Mike Baskey, IBM
605 Alvin Black, CA
606 Michel Drescher, Fujitsu Limited
607 Vivian Lee, Fujitsu Limited
608 Paul Lipton, CA
609 Yasuhide Matsumoto, Fujitsu Limited
610 Marcelo Perazolo, IBM
611 Abdi Salahshour, IBM
612 David Snelling, Fujitsu Limited
613 Jeffrey Vaught, CA

Appendix B. Revision History

Revision	Date	Editor	Changes Made
Wd-01	2009/11/12	Vivian Lee	Created the initial working draft by converting the input specification to OASIS template.
Wd-02	2010/05/08	Stavros Isaiadis	Added Types Store text. Added Appendix B for resource model and possible REST implementation Modified Interface section Replaced XPath with XQuery where necessary Removed the specification URIs and version info as this is only a working draft at the moment Replaced "Autonomic" with "Automation"
Wd-03	2010/09/22	Stavros Isaiadis	Preparing for CD approval, so kept only interface changes and removed Types Store and REST appendix as immature for CD at this point.
Wd-04	2010/09/27	Stavros Isaiadis	Polished for CD preparation (accepted/rejected changes as per discussions, etc.)
CD-01	2010/10/05	Stavros Isaiadis	Modified headers to denote CD status
CD-01 Rev 01	2011/03/21	Stavros Isaiadis	Added related symptoms and incident to the symptom element. Minor other changes.
CD-01 Rev 03	2011/05/06	Stavros Isaiadis	Changes in associated protocols and protocol groups
CD-01 Rev 04	2011/05/09	Jeff Vaught	Added ProtocolGroup and Incident ID. Some cleaning up of the schemas.
CD-01 Rev 05	2011/06/13	Stavros Isaiadis	Cleaning up. Made PotentialSyndromes a structured collection
CD-01 Rev 06	2011/06/27	Jeff Vaught	Changed <xsd:any> to xsd:any, as it is not an element. Cleaned up ProtocolGroup definition.
CD-01 Rev 07	2011/08/29	Jeff Vaught	Added SymptomType and ProtocolType sections along with their pseudoschemata.
CD-01 Rev 08	2011/08/30	Stavros Isaiadis	Added interfaces and some text for the SymptomType and PrescriptionType. Minor fixes.
CD-01 Rev 09	2011/09/19	Jeff Vaught	Added comments/changes per 9/19 review meeting.
CD-01 Rev 13	2011/10/21	Stavros Isaiadis	Added extensibility text; added example of

			Catalogue authoring; other minor changes throughout
CD-01 Rev 14	2011/10/21	Stavros Isaiadis	Added examples for each information model element
CD-01 Rev 15	2011/10/21	Stavros Isaiadis	Harmonized enumeration types, modifications in the examples and Appendix B
CD-01 Rev 16	2011/11/06	Jeff Vaught	Minor organization changes, modifications to pseudo xml examples, and section 5.2 diagram.
CD-01 Rev 17	2011/11/21	Jeff Vaught	Tidying of table widths, include missing label in 5.2.
CD-02 Rev 01	2011/11/22	Jeff Vaught	Initial CD-02
CD-02 Rev 02	2012/07/30	Stavros Isaiadis	Changes as per admin comments on CD-01; minor other modifications
CD-02 Rev 03	2012/10/09	Stavros Isaiadis	Minor changes to prepare for voting

616