Symptoms Automation Framework (SAF) Version 1.0

Committee Specification Draft ~~02~~03 / Public Review Draft ~~01~~02

~~13 December 2011~~

29 October 2012

Specification URIs

This version:

~~N/A~~
http://docs.oasis-open.org/saf/saf/v1.0/csprd02/saf-v1.0-csprd02.doc (Authoritative)
http://docs.oasis-open.org/saf/saf/v1.0/csprd02/saf-v1.0-csprd02.html
http://docs.oasis-open.org/saf/saf/v1.0/csprd02/saf-v1.0-csprd02.pdf

Previous version:

~~http://www.oasis-open.org/committees/download.php/39685/SAF-Specification-CD_01.doc~~
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:

~~N/A~~
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)~~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/csprd02/schemas/

Declared XML namespace:

- ~~http://docs.oasis-open.org/saf/ns/symptoms/2011/10~~
- 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 ~~a normative set of schema and WSDL documents~~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. ~~13 December 2011.~~29 October 2012. OASIS Committee Specification Draft ~~02~~03 / Public Review Draft ~~01.~~02. http://docs.oasis-open.org/saf/saf/v1.0/csprd02/saf-v1.0-csprd02.html.

Notices

Table of Contents

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/~~2011/10~~2012/07

1.3 Normative References

[RFC2119]	S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.
[XPATH20]	A. Berglund et al, XML Path Language, version 2.0, http://www.w3.org/TR/xpath20/, January 2007.
[XQUERY]	S. Boag et al, XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery/, January 2007.
[XML10]	T. Bray et al, Extensible Markup Language (XML) 1.0, November 2008.

~~NOTE: The proper format for a citation to an OASIS Technical Committee's work (whether Normative or Non-Normative) is:~~

~~OASIS~~

~~Stage (Committee Draft 01, Committee Draft 02, Committee Specification 01, etc. or Standard)~~

~~Title (italicized or in quotation marks)~~

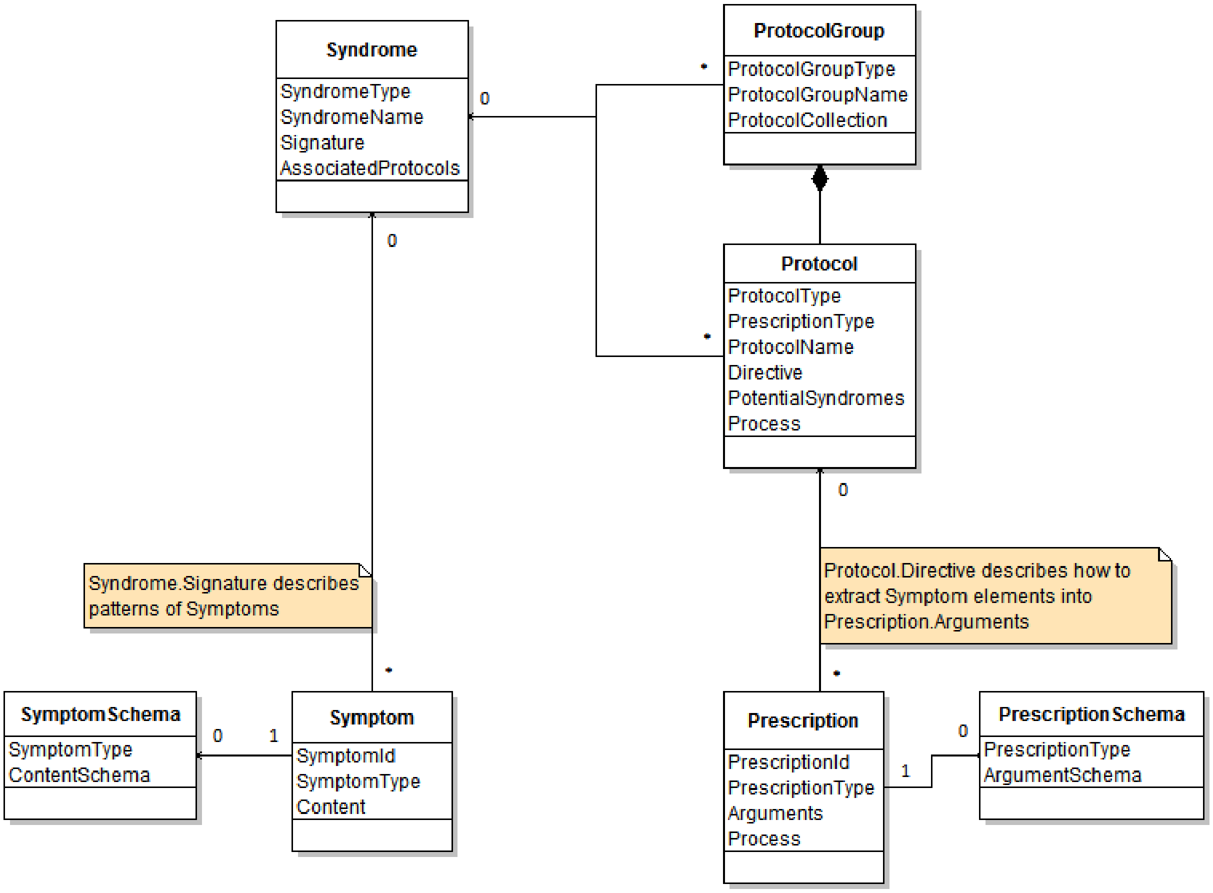
~~Approval Date (Month YYYY)~~

~~URI of the actual Authoritative Specification (namespace is not acceptable as the content changes over time)~~

~~For example:~~

~~EDXL-HAVE~~	~~OASIS Standard, "Emergency Data Exchange Language (EDXL) Hospital AVailability Exchange (HAVE) Version 1.0", November 2008.~~ ~~http://docs.oasis-open.org/emergency/edxl-have/os/emergency_edxl_have-1.0-spec-os.doc~~

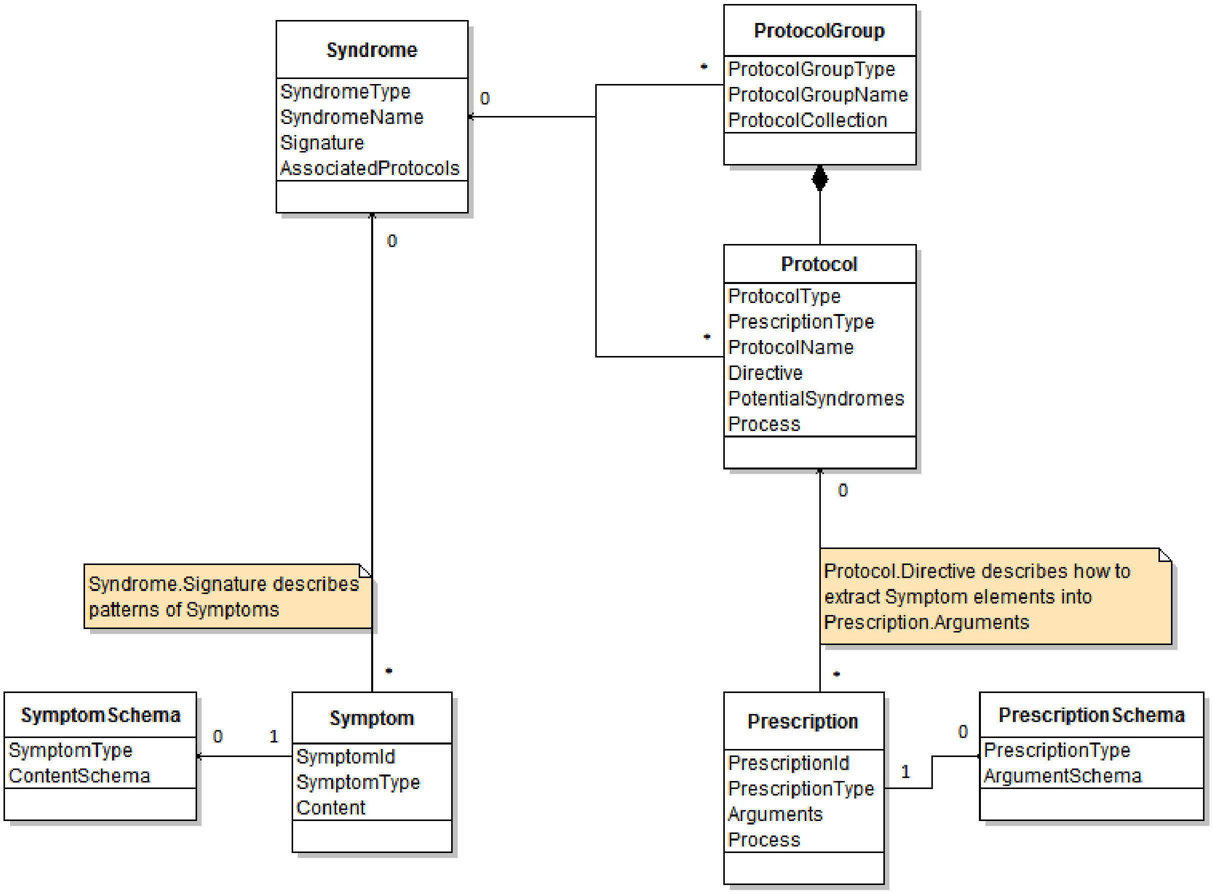
60
2 Information Model



61

62

~~saf:~~



2.1 SAFType

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

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.

2.1.1 Non Normative Pseudo Schema

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

```
<~~saf:~~SAFType>
  <Uri>xsd:anyURI</Uri>
  <Version>xsd:string</Version>?
</~~saf:~~SAFType>
```

Example of a SAFType for a Fever Syndrome:

```
<~~saf:~~SAFType>
  <Uri>http://~~saf~~example.com/saf/types/syndromes/fever/</Uri>
  <Version>1</Version>
</~~saf:~~SAFType>
```

2.2 Syndrome

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

Field	Type	Properties	Description
Syndrome Type	~~saf:~~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] ProtocolGroup Reference[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 more Protocols and/or zero or more groups of Protocols. Protocols within a group are executed together.

2.2.1 Non Normative Pseudo Schema

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

```
<Syndrome>
  <SyndromeType>saf:SAFType</SyndromeType>
  <SyndromeName>xsd:string</SyndromeName>
  <Description>xsd:string</Description>
  <Likelihood>
    [Common|Uncommon|Rare|NotAvailable]
  </Likelihood>
  <Impact>
    [HighImpact|ModerateImpact|LowImpact|UnknownImpact]
  </Impact>
  <Urgency>
    [HighUrgency|ModerateUrgency|LowUrgency|UnknownUrgency]
  </Urgency>
  <Signature>xsd:string</Signature>
  <AssociatedProtocols>
    <ProtocolReference>saf:SAFType</ProtocolReference>*
    <ProtocolGroupReference>saf:SAFType</ProtocolGroupReference>*
  </AssociatedProtocols>?
</Syndrome>
```

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

```
<Syndrome>
  <SyndromeType>
    ~~<Uri>http://saf.com/types/syndromes/fever/</Uri>~~
    <Uri>http://example.com/saf/types/syndromes/fever/</Uri>
    <Version>2</Version>
  </SyndromeType>
  <SyndromeName>FeverSyndrome</SyndromeName>
  <Description>Syndrome identifying fever</Description>
  <Likelihood>Common</Likelihood>
  <Impact>Low</Impact>
  <Urgency>Moderate</Urgency>
  <Signature>
    for $x in /Symptoms/Symptom
    where

$x[SymptomType="http://~~saf~~example.com/saf/types/symptoms/temperature/"]
      and $x/Content/Temperature[Value >= 38]
    return $x
  </Signature>
  <AssociatedProtocols>
    <ProtocolReference>
~~      <Uri>http://saf.com/types/protocols/aspirin/</Uri>~~
      <Uri>http://example.com/saf/types/protocols/aspirin/</Uri>
    </ProtocolReference>
    <ProtocolGroupReference>
~~      <Uri>http://saf.com/types/protocol-groups/diagnosefever/</Uri>~~
      <Uri>
          http://example.com/saf/types/protocol-groups/diagnosefever/
      </Uri>
    </ProtocolGroupReference>
  </AssociatedProtocols>
</Syndrome>

```

2.3 Protocol

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

Field	Type	Properties	Description
Protocol Type	~~saf:~~SAFType	Required, Immutable	ProtocolType uniquely defines the semantics of the Protocol.
Prescription Type	~~saf:~~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.

2.3.1 Non Normative Pseudo Schema

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

```
<Protocol>
  <ProtocolType>saf:Type</ProtocolType>
  <PrescriptionType>saf:Type</PrescriptionType>
  <ProtocolName>xsd:string</ProtocolName>
  <Description>xsd:string</Description>
  <Effectiveness>
    [Effective|PartiallyEffective|
     BestEffort|Ineffective|Inconclusive]
  </Effectiveness>
  <Risk>
    [HighRisk|ModerateRisk|LowRisk|UnknownRisk]
  </Risk>
  <Duration>
    [LongDuration|ModerateDuration|ShortDuration|UnknownDuration]
  </Duration>
  <Function>
    [Diagnostic|Preventative|Remedial|
     Diagnostic_Preventative|Diagnostic_Remedial|
     Preventative_Remedial|Diagnostic_Preventative_Remedial|Other]
  </Function>
  <Directive>xsd:string</Directive>
  <PotentialSyndromes>
    <SyndromeReference>saf:SAFType</SyndromeReference>*
  </PotentialSyndromes>?
  <Process>xsd:string</Process>?
</Protocol>
```

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

```
<Protocol>
  <ProtocolType>
    ~~<Uri>http://saf.com/types/protocols/aspirin/</Uri>~~
    <Uri>http://example.com/saf/types/protocols/aspirin/</Uri>
  </ProtocolType>
  <PrescriptionType>
    <Uri>
    ~~  http://saf.com/types/prescriptions/aspirin/~~
      http://example.com/saf/types/prescriptions/aspirin/
    </Uri>
  </PrescriptionType>
  <ProtocolName>AspirinProtocol</ProtocolName>
  <Description>Medication for Fever</Description>
  <Effectiveness>BestEffort</Effectiveness>
  <Risk>Low</Risk>
  <Duration>Short</Duration>
  <Function>Remedial</Function>
  <Directive>
    for $x in /Symptoms/Symptom
      let $subject := $x/Subject
      let $value := fn:number($x/Content/Temperature/Value)
      return
      <Details>
        <Subject>$subject</Subject>
        (: Give 1 aspirins for every 2 degrees above 38 :)
        <AspirinCount>
```

```
          {if ($value > 38) then (
             fn:floor($value - 38) div 2)
           ) else (0)}
        </AspirinCount>
      </Details>
  </Directive>
  <PotentialSyndromes>
    <SyndromeReference>
      <Uri>
    ~~    http://saf.com/types/syndromes/fever/~~
        http://example.com/saf/types/syndromes/fever/
      </Uri>
    </SyndromeReference>
  </PotentialSyndromes>
  <Process>
    ProvisionAspirin(Subject, AspirinCount);
  </Process>
</Protocol>
```

2.4 ProtocolGroup

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

Field	Type	Properties	Description
Protocol Group Type	~~saf:~~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.

2.4.1 Non Normative Pseudo Schema

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

```
<ProtocolGroup>
  <ProtocolGroupType>saf:SAFType</ProtocolGroupType>
  <ProtocolGroupName>xsd:string</ProtocolGroupName>
  <Description>xsd:string</Description>
  <ProtocolCollection>
    <ProtocolReference>saf:SAFType</ProtocolReference>+
  </ProtocolCollection>
</ProtocolGroup>
```

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

```
<ProtocolGroup>
  <ProtocolGroupType>
~~    <Uri>http://saf.com/types/protocol-groups/diagnosefever/</Uri>~~
    <Uri>http://example.com/saf/types/protocol-groups/diagnosefever/</Uri>
  </ProtocolGroupType>
  <ProtocolGroupName>FeverDiagnosis</ProtocolGroupName>
  <Description>
    Actions necessary to diagnose the type of fever.
  </Description>
  <ProtocolCollection>
    <ProtocolReference>
~~      <Uri>http://saf.com/types/protocols/blood_culture/</Uri>~~
      <Uri>http://example.com/saf/types/protocols/blood_culture/</Uri>
    </ProtocolReference>
    <ProtocolReference>
~~      <Uri>http://saf.com/types/protocols/skin_temperature/</Uri>~~
      <Uri>http://example.com/saf/types/protocols/skin_temperature/</Uri>
    </ProtocolReference>
  </ProtocolCollection>
</ProtocolGroup>
```

2.5 Prescription

A Prescription is an instance of a process, which MAY correspond to a Protocol. It is used to provide remediation, diagnostics, preventative measures, or optimization to be performed. Prescriptions MAY represent automated or Manual steps. A Prescription includes arguments specific to the subject or 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	~~saf:~~SAFType	Required, Immutable	The PrescriptionType defines the semantics of this Prescription. All Prescriptions baring 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.

2.5.1 Non Normative Pseudo Schema

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

```
<Prescription>
  <PrescriptionId>xsd:anyURI</PrescriptionId>
  <PrescriptionType>saf:SAFType</PrescriptionType>
  <ExpirationDate>
    xsd:dateTime
  </ExpirationDate>?
  <Arguments>xsd:any</Arguments>?
  <Process>xsd:string</Process>?
</Prescription>
```

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

```
<Prescription>
  <PrescriptionId>
    ~~http://saf.com/prescriptions/aspirin/12345~~
    http://example.com/saf/prescriptions/aspirin/12345
  </PrescriptionId>
  <PrescriptionType>
    ~~<Uri>http://saf.com/types/prescriptions/aspirin/</Uri>~~
    <Uri>http://example.com/saf/types/prescriptions/aspirin/</Uri>
    <Version>2</Version>
  </PrescriptionType>
  <ExpirationDate>2011-10-23</ExpirationDate>
  <Arguments>
    <Details>
      <Subject>http://~~saf~~example.com/saf/subjects/patient-234</Subject>
      <AspirinCount>2</AspirinCount>
    </Details>
  </Arguments>
  <Process>
    ProvisionAspirin(Subject, AspirinCount);
  </Process>
</Prescription>
```

2.6 Symptom

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

Symtoms may be linked to other previously emitted symptoms by specifying the unique ID of those symptoms and the type of relationship to them (e.g. causal, supersedes, custom, etc). Symptoms may 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	~~saf:~~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.

2.6.1 Non Normative Pseudo Schema

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

```
<Symptom>
  <SymptomId>xsd:anyURI</SymptomId>
  <SymptomType>saf:SAFType</SymptomType>
  <CreationDate>xsd:dateTime</CreationDate>
  <Confidence>
    [HighConfidence|ModerateConfidence|
     LowConfidence|UnknownConfidence]
  </Confidence>
  <Reporter>xsd:anyURI</Reporter>
  <Subject>xsd:anyURI</Subject>
  <ExpirationDate>
    xsd:DateTime
  </ExpirationDate>?
  <RelatedSymptoms>
    <RelatedSymptom type=”[Causal|Supersedes|Repetition|Other]”>
      xsd:anyURI
    </RelatedSymptom>+
  </RelatedSymptoms>?
  <Incident>xsd:anyURI</Incident>?
  <Content>xsd:any</Content>?
</Symptom>
```

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

```
<Symptom>
  <SymptomId>
    ~~http://saf.com/symptoms/temperature/2~~
    http://example.com/saf/symptoms/temperature/2
  </SymptomId>
  <SymptomType>
    ~~<Uri>http://saf.com/types/symptom/temperature/</Uri>~~
    <Uri>http://example.com/saf/types/symptom/temperature/</Uri>
  </SymptomType>
  <CreationDate>2011-10-24 13:10:05</CreationDate>
  <Confidence>High</Confidence>
  <Reporter>http://~~saf~~example.com/saf/reporters/tempsensor-123/</Reporter>
  <Subject>http://~~saf~~example.com/saf/subjects/patient-234/</Subject>
  <ExpirationDate>2011-10-24 14:10:05</ExpirationDate>
  <RelatedSymptoms>
    <RelatedSymptom type=”Supersedes”>
      http://~~saf~~example.com/saf/symptoms/temperature/1
    </RelatedSymptom>
  </RelatedSymptoms>
  <Incident>http://~~saf~~example.com/saf/incidents/12345</Incident>
  <Content>
    <Temperature>
      <Value>38</Value>
      <Scale>C</Scale>
    </Temperature>
  </Content>
</Symptom>
```

2.7 SymptomSchema

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

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

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

Field	Type	Properties	Description
Symptom Type	~~saf:~~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.

2.7.1 Non Normative Pseudo Schema

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

```
<SymptomSchema>
  <SymptomType>saf:SAFType</SymptomType>
  <ContentSchema>xsd:any</ContentSchema>
</SymptomSchema>
```

Example of a schema for temperature information.

```
<SymptomSchema>
  <SymptomType>
    ~~<Uri>http://saf.com/types/symptom/temperature/</Uri>~~
    <Uri>http://example.com/saf/types/symptom/temperature/</Uri>
  </SymptomType>
  <ContentSchema>
    <Temperature>
      <Value>xsd:float</Value>
      <Scale>[C|F]</Scale>
    </Temperature>
  </ContentSchema>
</SymptomSchema>
```

2.8 PrescriptionSchema

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

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

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

Field	Type	Properties	Description
Prescription Type	~~saf:~~SAFType	Required, Unique Immutable	The PrescriptionType defines the semantics of this PrescriptionSchema. All PrescriptionSchemas baring 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.

2.8.1 Non Normative Pseudo Schema

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

```
<PrescriptionSchema>
  <PrescriptionType>saf:SAFType</PrescriptionType>
  <ArgumentSchema>xsd:any</ArgumentSchema>
</PrescriptionSchema>
```

Example of a schema for the aspirin disposing Prescription.

```
<PrescriptionSchema>
  <PrescriptionType>
    ~~<Uri>http://saf.com/types/prescriptions/aspirin/</Uri>~~
    <Uri>http://example.com/saf/types/prescriptions/aspirin/</Uri>
    <Version>2</Version>
  </PrescriptionType>
  <ArgumentSchema>
    <Details>
      <Subject>xsd:anyURI</Subject>
      <AspirinCount>xsd:integer</AspirinCount>
    </Details>
  </ArgumentSchema>
</PrescriptionSchema>
```

3 Architectural Roles

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

~~2.9~~3.1 Information Sources

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

~~2.9.1~~3.1.1 Syndrome and Protocol Catalog

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

~~2.9.2~~3.1.2 Symptom Store

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

~~2.10~~3.2 Active Roles

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

~~2.10.1~~3.2.1 Catalog Source

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

~~2.10.2~~3.2.2 Symptom Source

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

~~2.10.3~~3.2.3 Case Manager

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

~~2.10.4~~3.2.4 Diagnostician

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

~~2.10.5~~3.2.5 Practitioner

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

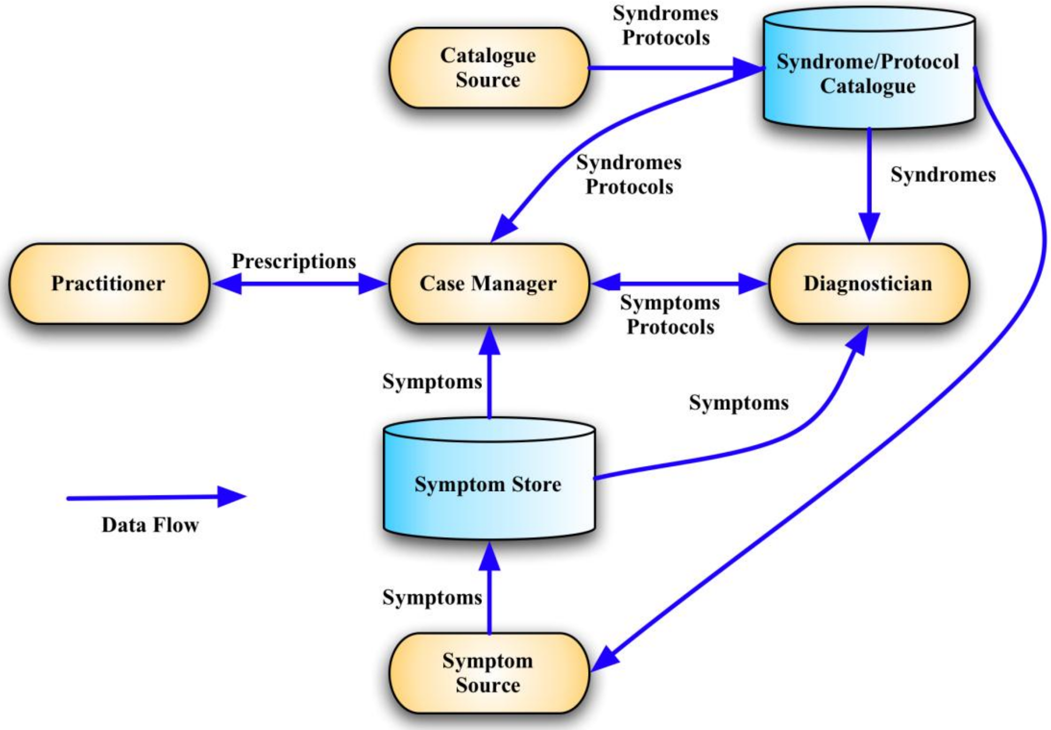


Figure 1: Roles and Information Stores in the Symptoms Automation Framework.

~~3~~4 Interfaces

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

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

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

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

Interface	Description	Candidate Role
Symptom Emitter	This is for the symptom sources or reporters emitting symptoms	Symptom Source
	Operations	
	• List supported types (Optional)	
Symptom Handler	This is for the entity that receives symptoms for further processing	Diagnostician Symptom Source Case Manager Others
	• Get a Symptom • Add a Symptom • Query Symptoms	
Prescription Emitter	The source for emitting a prescription	Case Manager
	N/A	
Prescription Handler	This is for component that receives and acts on the prescription	Practitioner Case Manager Others
	• Receive Prescription	

	• List supported types

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

Catalog Handler	This for the component that is capable of handling specific syndromes and protocols.	Catalog Source Case Manager Others
	• 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	

45Notes on Future Specification Development

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

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

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

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

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

~~5~~6 Examples

~~5.1~~6.1 Medical Sequence Diagram

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

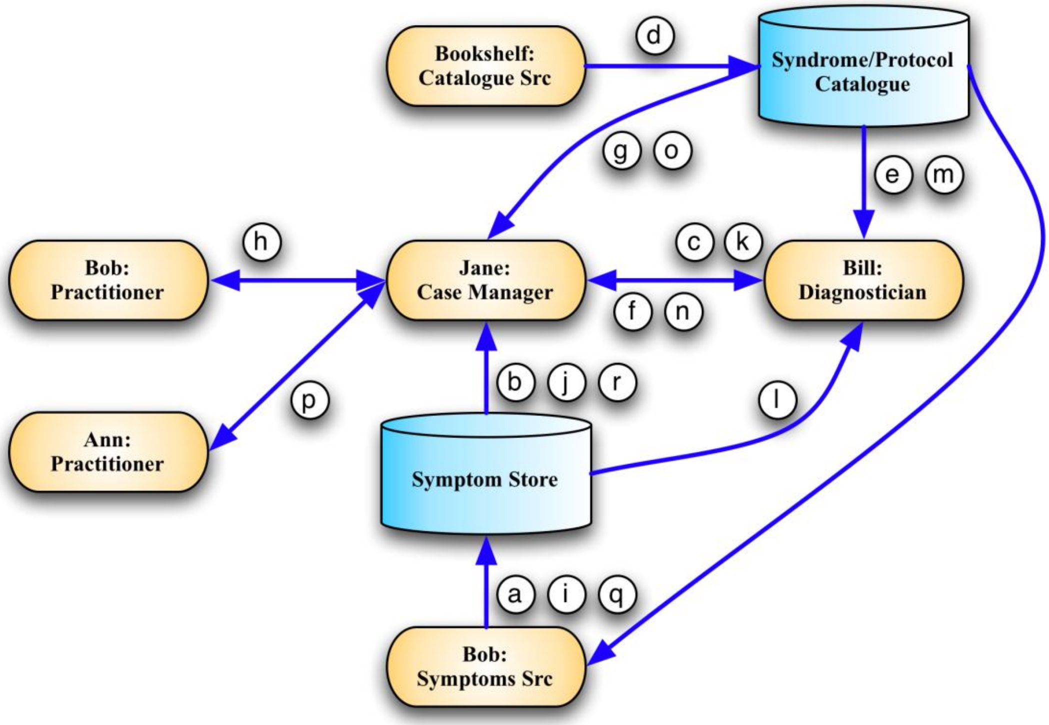


Figure 2: Medical Diagnosis Sequence

Symptoms Process:

a) Bob (Symptoms Source) says, "I don't feel so good." (Symptom).

b) Jane (Case Manager) hears of Bob's Symptom, and

c) asks Bill (Diagnostician), "What do you think it is? "

d) Bill collects a first aid book (Catalog) from the bookshelf (Catalog Source).

e) Bill consults the Catalog and learns that the top entry listing the "I don't feel so good" Symptom is a "Fever" (Syndrome), and

f) he passes this to Jane.

g) Jane looks up "Fever" in the Catalog where it recommends, "take temperature" (Protocol) to verify the Syndrome, using a mercury thermometer (Prescription).

h) She then instructs Bob (this time a Practitioner) to take his own temperature.

i) Bob reports his temperature (a new Symptom).

j) Jane reads it and

k) again consults Bill.

l) Bill reads the value of the temperature and

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

~~5.2~~6.2 Catalogue Authoring Diagram

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

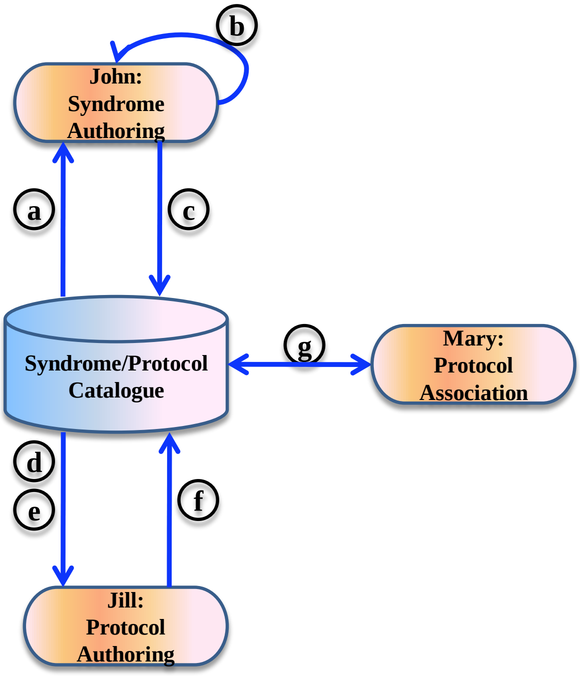


Figure 3: Catalog Authoring

Authoring Process:

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

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

e) Jill also needs to know how to extract these arguments, so she looks into the Fever Syndrome’s Signature to find out what it will return as a result.

f) Jill then creates a Protocol with a Directive able to generate the above PrescriptionSchema by extracting Subject and AspirinCount information from the Symptoms returned by the Syndrome signature. She adds this Protocol to the Catalogue.

g) Jill then goes on to associate this Protocol to the Fever Syndrome.

~~6~~7 Conformance

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

Normative text within this specification takes precedence over normative outlines, which in turn take precedence over the XML Schema [XML Schema Part 1, Part 2] and WSDL [WSDL 1.1] descriptions, which in turn take precedence over examples.

Appendix A. Acknowledgements

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

Participants:

Mike Baskey, IBM
Alvin Black, CA
Stavros Isaiadis, Bank of America Merrill Lynch (previously Fujitsu Limited)
Vivian Lee, Fujitsu Limited
Paul Lipton, CA
Yasuhide Matsumoto, Fujitsu Limited
Marcelo Perazolo, IBM
David Snelling, Fujitsu Limited
Jeffrey Vaught, CA

Co-Developers of the initial contributions:

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

Mike Baskey, IBM
Alvin Black, CA
Vivian Lee, Fujitsu Limited
Paul Lipton, CA
Yasuhide Matsumoto, Fujitsu Limited
Marcelo Perazolo, IBM
Abdi Salahshour, IBM
David Snelling, Fujitsu Limited
Jeffrey Vaught, CA

Acknowledgements of the initial contributions:

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

Mike Baskey, IBM
Alvin Black, CA
Michel Drescher, Fujitsu Limited
Vivian Lee, Fujitsu Limited
Paul Lipton, CA
Yasuhide Matsumoto, Fujitsu Limited
Marcelo Perazolo, IBM
Abdi Salahshour, IBM
David Snelling, Fujitsu Limited
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