A. Introduction

The Service Tag Listener makes Service Tag information available on the network. It also accepts messages indicating that the state of a Service Tag has changed--such as when a product instance is registered with its provider (such as Sun). By default, the stlisten daemon will listen for information retrieval and update messages (using an HTTP-based protocol) on TCP port 4950.

The communication protocol are intended to be the default, but not exclusive mechanisms for exposing Service Tag information to a network. In many cases, it will be desirable to use management protocols (such as SNMP or WBEM) in addition to or instead of the default. Service Tags will support such choices wherever possible. Those alternatives are not being adopted as the default simply because they are heavier-weight (being generalized by design) and thus would be difficult to universally support on the variety of target platforms for Service Tags (everything from full-OS instances of Solaris, Linux, HP-UX, and Windows to minimal OS instances for embedded service processors).

Note that in most cases, the listener software is expected to be running with the "noaccess" uid and gid. This ensures it access to read and write any entry in the local machine's Service Tag registry, but does not provide any special privileges unrelated to Service Tags.

The following sections describe the default communication protocol.

B. Communication Protocol

The built-in communications protocol for Service Tag listeners uses HTTP messages over TCP (on port 4950, by default). Requests (usually from a Service Tag "Collector" component) must be unicast. The format for communication messages (and the expected responses) are described in this section.

B1. GET /stv1/agent/ HTTP/1.0

or

GET /stv1/agent/${agent_urn} HTTP/1.0

Purpose

Used to get information about this agent. Since the first URI ("/stv1/agent/") would be well-known and non-variable across agents, it would generally be used as part of an initial query from a collector. The second URI ("/stv1/agent/${agent_urn}") would obviously vary for each agent and thus would only be useful to collectors which already have some specific knowledge of the current agent.

Response Description

Returns the XML representation of this agent's state (in the same "v1" format used between the collector and catcher).

```
Example Response
----------------
 HTTP/1.1 200 OK
 Content-type: text/xml;charset=UTF-8

 <?xml version="1.0" encoding="UTF-8"?>
 <agent>
   <agent_urn>urn:stagent:123-456</agent_urn>
   <system_info>
   <system>SunOS</system>
   <host>sr1-ubrm-26</host>
   <release>5.10</release>
   <machine>sun4v</machine>
   <architecture>sparc</architecture>
   <platform>SUNW,Sun-Fire-T200</platform>
   <manufacturer>Sun_Microsystems</manufacturer>
   <cpus_installed>32</cpus_installed>
   <memory_installed>32760</memory_installed>
   </system_info>
 </agent>
```

B2. GET /stv1/svctag/ HTTP/1.0

```
Purpose
-------
 Used to obtain a list of svctags visible via this agent (and the
 URLs with which each may be accessed).

Response Description
--------------------
 Returns a list specific URLs for the svctags visible via this agent.

Example Response
----------------
 HTTP/1.1 200 OK
 Content-type: text/xml;charset=UTF-8

 <service_tags>
   <link type="service_tag"
    href="/stv1/svctag/urn:st:0087a220-1dd2-11b2-8512-8400200ed56e" />
   <link type="service_tag"
    href="/stv1/svctag/urn:st:007bf510-1dd2-11b2-81ee-8400200ed56e" />
 </service_tags>
```

B3. GET /stv1/svctag/${svctag_urn} HTTP/1.0

```
Purpose
-------
 Used to get information about a specific svctag.

Response Description
--------------------
 Returns the XML representation of a specific svctag's state (in the same
 "v1" format used between the collector and catcher).  For example:

Example Response
----------------
```

```
HTTP/1.1 200 OK
Content-type: text/xml;charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<service_tag>
  <instance_urn>urn:st:0087a220-1dd2-11b2-8512-8400200ed56e</instance_urn>
  <product_name>Sun Java System Calendar Server 6.3</product_name>
  <product_version>6.3</product_version>
  <product_urn>urn:uuid:8b984eb8-c43b-11da-857a-080020a9ed93</product_urn>
  <product_parent>javaes</product_parent>
  <product_vendor>Sun</product_vendor>
  <platform_arch>sparc</platform_arch>
  <status>UNREGISTERED</status>
  <timestamp>2006-12-19 17:54:41 GMT</timestamp>
  <customer_asset_tag/>
  <group_id/>
  <container>global</container>
  <source>demo-script</source>
  <installer_uid>86797</installer_uid>
</service_tag>
```

B4. PUT /stv1/agent/ HTTP/1.0

or

PUT /stv1/agent/${agent_urn} HTTP/1.0

```
Purpose
-------
 Used to update the state of this agent (such as marking it as having
 been registered with Sun).

Example Response (success)
--------------------------
 <response>
   <major-code>200</major-code>
   <minor-code>0</minor-code>
   <message>agent urn:stagent:123-456 updated</message>
   <timestamp>2006-11-21 11:38:56 MST</timestamp>
 </response>

Example Response (failure)
--------------------------
 <response>
   <major-code>409</major-code>
   <minor-code>2</minor-code>
   <message>unknown agent: urn:stagent:123-456</message>
   <timestamp>2006-11-21 11:38:56 MST</timestamp>
 </response>
```

B5. PUT /stv1/svctag/${svctag_urn} HTTP/1.0

```
Purpose
-------
 Used to update the state of a specific svctag (such as marking it as
 having been registered with Sun).
```

Example Response (success)

```
<response>
  <major-code>200</major-code>
  <minor-code>0</minor-code>
  <message>
   svctag urn:st:00342294-1dd2-11b2-ba36-8400200ed56e updated
  </message>
  <timestamp>2006-11-21 11:38:56 MST</timestamp>
</response>
```

Example Response (failure)

```
<response>
  <major-code>409</major-code>
  <minor-code>2</minor-code>
  <message>
   unknown svctag: urn:st:00342294-1dd2-11b2-ba36-8400200ed56e
  </message>
  <timestamp>2006-11-21 11:38:56 MST</timestamp>
</response>
```

B6. <all other HTTP requests>

Purpose

Messages which are valid HTTP requests but do not correspond to a meaningful svctag-related message should be answered with appropriate HTTP response codes. For example, POST and OPTIONS requests would likely receive HTTP 405 (Method Not Allowed) responses. A GET request referencing a non-existent URI would likely receive an HTTP 404 (Not Found) response. And a PUT request referencing a non-existent svctag or agent URN or URI would likely receive a HTTP 409 (Conflict with Current State) response.