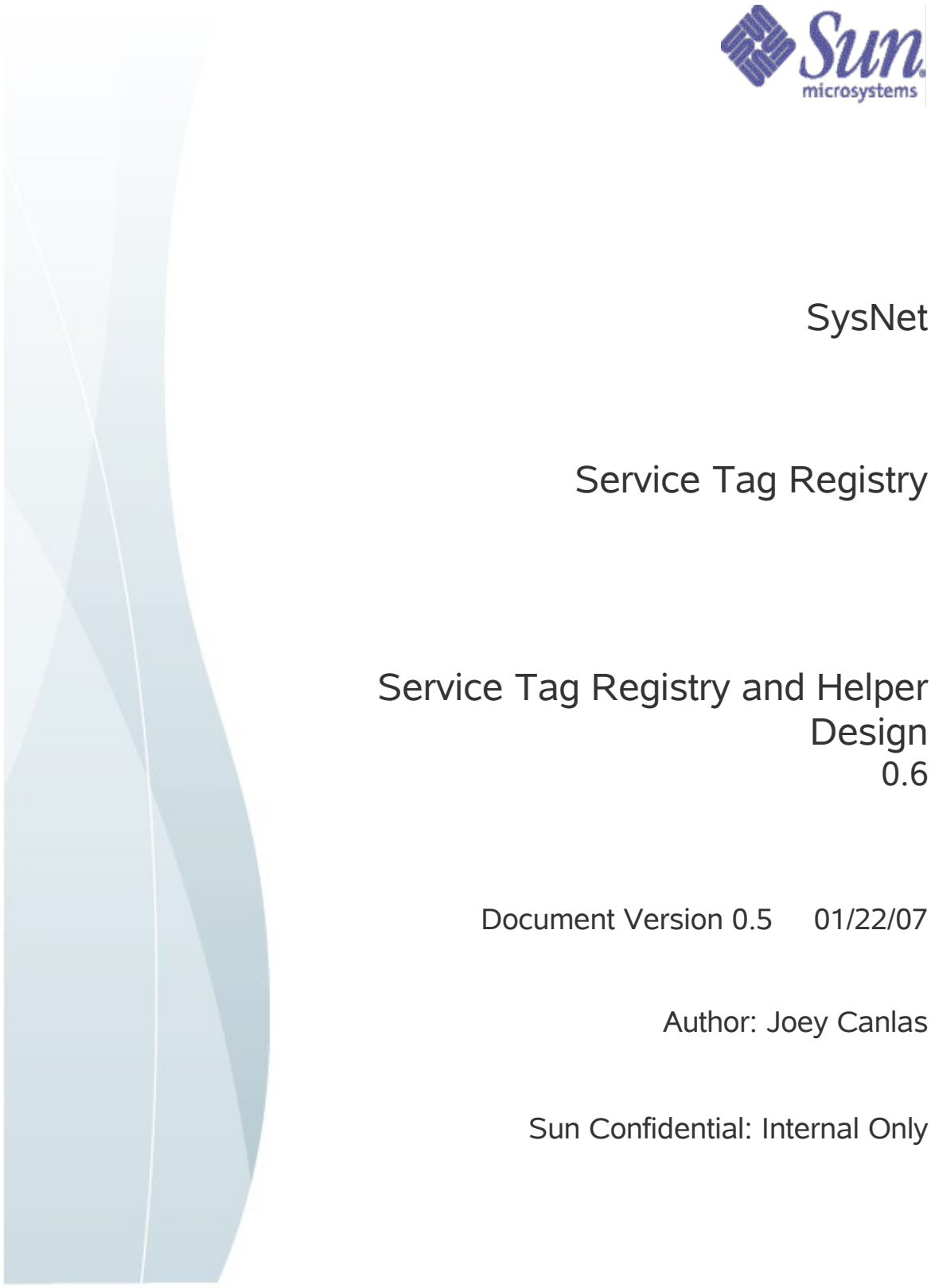


SysNet

Service Tag Registry

Service Tag Registry and Helper Design
0.6

Document Version 0.5 01/22/07

Author: Joey Canlas

Revision History

Document Version	Comments	Date (mm/dd/yy)	Author
0.1	Initial Draft	11/30/06	Joey Canlas
0.2	Updated schema, API and CLI based on team review	12/04/06	Joey Canlas
0.3	Added sample CLI interactive session – appendix D.	12/07/06	Joey Canlas
0.4	Added new fields in registry schema and other changes based on SysNet review	12/15/06	Joey Canlas
0.5	Added product parent identifier and product defined identifier based on comments from Martin Mayhead.	12/22/06	Joey Canlas
0.6	Changed DTD location and ST user/group based on PSARC review.	01/22/07	Joey Canlas

Document Purpose and Scope

The purpose of this document is to describe the Service Tag Registry and its API and CLI specification.

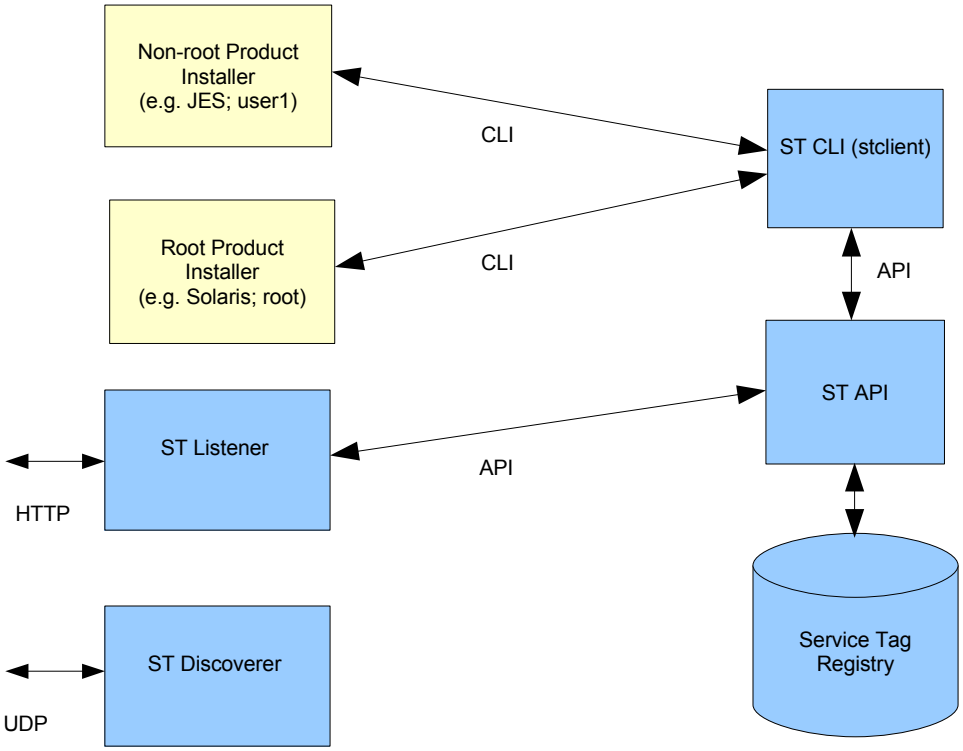
Table of Contents

1 Functional Overview

The Service Tag Registry is a local repository containing a list of the product instances installed in the system. Service tags are created by Sun product teams when their products are installed in the system. Each service tag has a unique instance identifier which is generated or supplied by the product team when the record is added to the registry. This identifier is used as the key when updating the record.

Service tags are eventually retrieved and updated by the Service Tag Agent. Collected service tags are sent to Sun for data mining and other business purposes. The agent may also perform product registration using information from the registry.

There are two (2) interfaces to the Registry: C API and Command line interface (CLI). These interfaces allow the authorized users to add, update, delete and extract the service tags in the registry. The interfaces provide record level security. Only root~~svctag ,~~ and the creator of the service tag can update, uninstall or delete the record. The diagram below how different clients interface with the registry. The stclient CLI is a public interface which can be called by root and non-root users. This binary is owned by ~~the user svctag~~ noaccess:noaccess and runs with an owner setuid. The C API is a private interface that interacts directly with the registry.



2 Service Tag Registry Details

The Service Tag directory is initially created when the genesis patch/package is installed. An example of the registry is shown in appendix C. The registry file is in XML format and is created the first time a service tag is registered. It is owned by user ~~svctag~~ noaccess with the group set to ~~svctag~~ noaccess. It has a permission of 664 (rw-rw-r--) and persisted in /var/sadm/servicetag/registry/servicetag.xml.

The Service Tag schema and DTD are included in the appendices A and B respectively. The registry is validated against the DTD everytime the registry is accessed. The DTD is persisted in /usr/share/lib/xml/dtd/servicetag.dtd and is installed as part of the Service Tag installation package. For development and testing, an alternate root directory may be specified. In this case, ~~The~~ the DTD should reside~~s~~ in the same directory as the registry. The XML schema is used for documentation purposes only. Validation using an XML schema requires a newer version of the libxml library which is not currently installed in older operating systems such as Solaris 8.

Registry Data Elements

Element	Description
instance_urn	Unique product instance identifier; generated when the service tag is added
product_name	Product name of the service tag
product_version	Product version of the service tag
product_urn	Sun Product unique identifier; swoRDFish id
product_parent_urn	Sun Product unique identifier, swoRDFish id, of the parent product
product_parent	Parent name of the product
product_defined_instance_id	Product team defined instance text that corresponds with the way that the product instance identifies itself in it's administration & management interfaces e.g. AS9.0 at 192.168.0.1
product_vendor	Manufacturer of the product
platform_arch	Platform architecture on which the product runs on, e.g. SPARC/Solaris
status	Status of the service tag. The valid values are UNREGISTERED, REGISTERED, UNINSTALLED, DEREGISTERED and REREGISTER.
customer_asset_tag	Textual notes on the asset that identifies the product instance in a way meaningful to the customer, e.g. Shopping Cart App Server #3
group_id	Group id which the product is associated
container	Text that indicates which container the product has been deployed, e.g. S10/zone1
source	Forensic marker to indicate the source of the service tag

Element	Description
installer_uid	User id who created the service tag. Set automatically when a record is added or updated.
timestamp	Date and timestamp in GMT when the service tag record was updated. This is set automatically. The format is YYYY-MM-DD HH24:MI:SS GMT.

3 CLI

One utility command, **stclient**, for adding, updating, deleting and displaying the service tag records will be provided. This command will be owned by ~~user svctag~~ noaccess:noaccess and have a permission of ~~755~~ 775 with the owner setuid enabled (u+s). This allows non-root users to add, update, delete and display service tags.

The CLI also provides an interactive command line interface. This mode is invoked when stclient is ran without any parameters. A sample of an interface session is shown in appendix D.

The stclient command will be written using **getopt** instead of getopt_long to parse the options. This will ensure compatibility with older O/S such as Solaris 8.

Command Options

Options	Description
No parameters	Interactive session. Invokes the interactive mode.
-x	Extract. Extracts and prints the contents of the Service Tag Registry in XML format. Valid operands: -r
-a	Add. Adds a service tag in the registry using the product name, version and URN as parameters. A unique product instance identifier is either supplied or generated and returned by the command. The status field is set to 'UNREGISTERED'. . Valid operands: -i -p -e -t -P -m -A -S -r
-u	Update. Updates a service tag in the registry using the product instance URN as the key. The fields status, may_contact and comment fields can be updated. Valid operands: -i -s -g -c -r
-d	Delete. Deletes a service tag in the registry using the product instance URN as the key. Valid operands are: -i -r
-f	Find. Finds and prints all the instance URN of a given product URN. Valid operand are: -t -r
-U	Uninstall. Uninstalls a service tag in the registry using the product instance URN as the key. If the current status of the service tag is UNREGISTERED, the service tag record is deleted. For all other status, the service tag is updated with a status of UNINSTALLED. Valid operands are: -i -r
-g	Get. Gets and prints the service tag fields of a given product instance URN. Valid operands are: -i -r
-h	Help. Displays the command options.

Options	Description
-v	Version. Displays the stclient version.

Command Operands

Operands	Description
-p <product>	Sets the product name of the service tag to be added, e.g. -p "Solaris 10 Operating System"
-e <product ver>	Sets the product version of the service tag to be added, e.g. -r "5.10"
-t <product URN>	Sets the Sun product unique identifier of the service tag to be added, e.g. -t urn:uuid:5005588c-36f3-11d6-9cec-fc96f718e113
-i <instance URN>	Sets the product instance unique identifier of the service tag to be updated, e.g. -i urn:uuid:3a444ab2-1dd2-11b2-a69d-830020782a6b
-F <parent URN>	Sets the Sun product unique identifier of the parent product
-P <parent product>	Sets the product parent name of the service tag to be added.
-I <product defined instance id>	Sets the product defined instance id
-m <vendor>	Sets the product vendor of the service tag to be added, e.g. Sun
-A <platform arch>	Sets the platform architecture of the service tag to be added. e.g. sparc
-s <status>	Sets the status of the service tag to be updated, e.g. -s REGISTERED. The valid status values are REGISTERED, UNREGISTERED, UNINSTALLED, DEREGISTERED, REREGISTER. The values are not case sensitive.
-c <customer asset tag>	Sets the customer notes about the asset.
-G <group id>	Sets the group id of the asset.
-z <container>	Sets the container of the service tag to be added.
-S <source>	Sets the source of the service tag to be added.
-r <root dir>	Sets the root directory where the command searches for the Registry. By default, the command searches /var/sadm/servicetag/registry. By setting this operand, the command will look for the Registry in the specified root directory, e.g. -a /home/user1 searches /home/user1/var/sadm/servicetag.

Summary of commands and their operands.

	Extract (-x)	Add (-a)	Update (-u)	Delete (-d)	Uninstall (-U)	Get (-g)	Find (-f)
Product instance URN (-i)		Optional	Required	Required	Required	Required	
Product name (-p)		Required					
Product version (-e)		Required					
Product URN (-t)		Required					Required
Product parent identifier (-F)		Optional	Optional				
Product parent (-P)		Required					
Product defined instance (-I)		Optional	Optional				
Product vendor (-m)		Required					
Platform arch (-A)		Required					
Status (-s)			Optional*				
Customer asset tag (-c)			Optional*				
Group Id			Optional*				
Container (-z)		Required					
Source (-S)		Required					
Root directory (-r)	Optional	Optional	Optional	Optional	Optional	Optional	Optional

Blank means not applicable. Optional* - at least one of these fields.

4 API

The Service Tag Helper C API provides a mechanism to access the registry. This is a private interface meant to be used by the Service software bundle. In general, the API does the following functions:

- Extract the registry
- Add service tags
- Update service tags
- Delete service tags
- Query the registry

A summary of the API and the error codes are shown in appendix E and F respectively. The API function calls return ST_ERR_NO_ERROR (0) if the operation was successful or the actual error code if it failed. The error code can be translated into a human readable format by calling st_error.

The API uses the libxml library (2.4.7 or higher) which is the XML C parser and toolkit developed for the Gnome project. Different versions of this library is included in various Solaris releases including Solaris 10. It is also available in different platforms such as Linux and Windows making the Service Tag API portable to these platforms. The libxml library supports file locking and thread safety.

5 Appendix A: Service Tag Schema

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="registry">
  <xs:attribute name="urn"     type="xs:string" use="required"/>
  <xs:attribute name="version" type="xs:string" use="required"/>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="service_tag" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="instance_urn"/>
            <xs:element ref="product_name"/>
            <xs:element ref="product_version"/>
            <xs:element ref="product_urn"/>
            <xs:element ref=”product_parent_urn”/>
            <xs:element ref="product_parent"/>
            <xs:element ref=”product_defined_inst_id”/>
            <xs:element ref="product_vendor"/>
            <xs:element ref="platform_arch"/>
            <xs:element ref="status"/>
            <xs:element ref="timestamp"/>
            <xs:element ref="customer_asset_tag"/>
            <xs:element ref="group_id"/>
            <xs:element ref="container"/>
            <xs:element ref="source"/>
            <xs:element ref="installer_uid"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

  <!-- definition of simple elements -->
  <xs:element name="instance_urn">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="255"/>
      </xs:restriction>
    </xs:simpleType>
   </xs:element>

  <xs:element name="product_name">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="255"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="product_version">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="63"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="product_urn">
    <xs:simpleType>
      <xs:restriction base="xs:string">
```

```
        <xs:minLength value="1"/>
        <xs:maxLength value="255"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="product_parent_urn">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="255"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="product_parent">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="255"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="product_defined_inst_id">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="255"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="product_vendor">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="63"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="platform_arch">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="63"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="status">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="UNREGISTERED"/>
        <xs:enumeration value="REGISTERED"/>
        <xs:enumeration value="UNINSTALLED"/>
        <xs:enumeration value="DEREGISTERED"/>
        <xs:enumeration value="REREGISTER"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="timestamp">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="24"/>
      </xs:restriction>
```

```
    </xs:simpleType>
  </xs:element>

  <xs:element name="customer_asset_tag">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="1023"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="customer_asset_tag">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="1023"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="group_id">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="63"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="container">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="63"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="source">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="63"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

  <xs:element name="installer_uid">
    <xs:simpleType>
      <xs:restriction base="xs:integer">
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

</xs:schema>
```

6 Appendix B: Service Tag Registry DTD

```
<!ELEMENT registry (service_tag*)>
<!ATTLIST registry urn CDATA #REQUIRED>
<!ATTLIST registry version CDATA #REQUIRED>
<!ELEMENT service_tag (instance_urn, product_name, product_version, product_urn,
product_parent_urn, product_parent, product_defined_inst_id, product_vendor,
platform_arch, status, timestamp, customer_asset_tag, group_id, container, source,
installer_uid)>
<!ELEMENT instance_urn       (#PCDATA)>
<!ELEMENT product_name       (#PCDATA)>
<!ELEMENT product_version    (#PCDATA)>
<!ELEMENT product_urn        (#PCDATA)>
<!ELEMENT product_parent_urn (#PCDATA)>
<!ELEMENT product_parent     (#PCDATA)>
<!ELEMENT product_defined_inst_id (#PCDATA)>
<!ELEMENT product_vendor     (#PCDATA)>
<!ELEMENT platform_arch      (#PCDATA)>
<!ELEMENT status             (#PCDATA)>
<!ELEMENT timestamp          (#PCDATA)>
<!ELEMENT customer_asset_tag (#PCDATA)>
<!ELEMENT group_id           (#PCDATA)>
<!ELEMENT container          (#PCDATA)>
<!ELEMENT source             (#PCDATA)>
<!ELEMENT installer_uid      (#PCDATA)>
```

7 Appendix C: Service Tag Registry Sample

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE registry SYSTEM
"~~var/sadm/servicetag/registry/~~/usr/share/lib/xml/dtd/servicetag.dtd">
<registry urn="urn:uuid:123450060" version="1.0">
  <service_tag>
    <instance_urn>urn:st:dae4a78a-1dd1-11b2-bc81-30020782a6b</instance_urn>
    <product_name>Solaris 10 Operating System</product_name>
    <product_version>5.10</product_version>
    <product_urn>urn:uuid:5005588c-36f3-11d6-9cec-c96f718e113</product_urn>
    <product_parent_urn></product_parent_urn>
    <product_parent>OS</product_parent>
    <product_defined_inst_id>Solaris 10 at 189.19.10.1</product_defined_inst_id>
    <product_vendor>Sun</product_vendor>
    <platform_arch>Sparc</platform_arch>
    <status>REGISTERED</status>
    <timestamp>2006-12-14 17:52:24 GMT</timestamp>
    <customer_asset_tag></customer_asset_tag>
    <group_id></group_id>
    <container>global</container>
    <source>demo-script</source>
    <installer_uid>91072</installer_uid>
  </service_tag>
</registry>
```

8 Appendix D: Service Tag CLI Interactive Session

```
~/helper/build/bin $ stclient
COMMANDS:
        h - Show this menu
        i - Initialize environment
        x - Extract service tag registry
        a - Add service tag
        u - Update service tag
        g - Get service tag
        d - Delete service tag
        U - Uninstall service tag
        f - Find service tags by product URN
        q - Quit
> i
Enter root directory or <enter> for system:
Environment initialized
Press enter to continue
> a
Enter instance URN (optional):
Enter product (e.g. Sun Web Server): Solaris 10 Operating System
Enter version (e.g. 6.1): 5.10
Enter product URN: urn:uuid:5005588c-36f3-11d6-9cec-c96f718e113
Enter product parent URN:
Enter product parent (e.g. JES): OS
Enter product defined instance id: Solaris 10 at 189.19.10.1Enter product vendor (e.g. Sun): Sun
Enter platform arch  (e.g. SPARC): sparc
Enter container (e.g. zone 0): global
Enter source (e.g. genesis patch): genesis patch
Solaris 10 Operating System 5.10 added
Product instance URN=urn:st:01c3ef08-1dd2-11b2-a55b-8400201d8c78
Press enter to continue
> x
<?xml version="1.0" encoding="UTF-8"?>
<registry>
  <version>1.0</version>
  <service_tag version=”1.0”>
    <instance_urn>urn:st:01c3ef08-1dd2-11b2-a55b-8400201d8c78</instance_urn>
    <product_name>Solaris 10 Operating System</product_name>
    <product_version>5.10</product_version>
    <product_urn>urn:uuid:5005588c-36f3-11d6-9cec-c96f718e113</product_urn>
    <product_parent_urn></product_parent_urn>
    <product_parent>OS</product_parent>
    <product_defined_inst_id>Solaris 10 at 189.19.10.1</product_defined_inst_id>
    <product_vendor>Sun</product_vendor>
    <platform_arch>sparc</platform_arch>
    <status>UNREGISTERED</status>
    <timestamp>2006-12-15 22:45:48 GMT</timestamp>
    <customer_asset_tag/>
    <group_id/>
    <container>global</container>
    <source>genesis patch</source>
    <installer_uid>91072</installer_uid>
  </service_tag>
</registry>

Press enter to continue
> u
Enter instance URN: urn:st:01c3ef08-1dd2-11b2-a55b-8400201d8c78
Enter status: registered
Enter customer asset tag: This is the customer notes
Enter group id: G1
Service tag urn:st:01c3ef08-1dd2-11b2-a55b-8400201d8c78 updated
Press enter to continue
> x
<?xml version="1.0" encoding="UTF-8"?>
<registry version=”1.0”>
  <service_tag>
    <instance_urn>urn:st:01c3ef08-1dd2-11b2-a55b-8400201d8c78</instance_urn>
    <product_name>Solaris 10 Operating System</product_name>
    <product_version>5.10</product_version>
    <product_urn>urn:uuid:5005588c-36f3-11d6-9cec-c96f718e113</product_urn>
    <product_parent_urn></product_parent_urn>
    <product_parent>OS</product_parent
    <product_defined_inst_id>Solaris 10 at 189.19.10.1</product_defined_inst_id>
    <product_vendor>Sun</product_vendor>
    <platform_arch>sparc</platform_arch>
    <status>REGISTERED</status>
    <timestamp>2006-12-15 22:46:20 GMT</timestamp>
    <customer_asset_tag>This is the customer notes</customer_asset_tag>
    <group_id>G1</group_id>
    <container>global</container>
    <source>genesis patch</source>
```

```
    <installer_uid>91072</installer_uid>
  </service_tag>
</registry>
```

9 Appendix E: Service Tag API

API	Description
int st_initialize(char *root_dir, st_env_t *)	Initializes the ST environment. This must be called prior to calling other functions.
void st_close(st_env_t *)	Cleans up the ST environment.
int st_is_initialized(st_env_t *)	Returns the initialization status of the ST environment.
int st_add(st_t *, st_env_t *)	Adds a service tag in the registry.
int st_update(st_t *, st_env_t *)	Updates a service tag in the registry.
int st_delete(char *instance_urn, st_env_t *)	Deletes a service tag in the registry.
int st_get(st_t *, st_env_t *)	Gets a service tag in the registry.
int st_uninstall(char *instance_urn, st_env_t *)	Uninstalls a service tag in the registry. Sets the status to 'UNINSTALLED'. In a case where the product was never registered, the entry is deleted.
int st_get_registry_xml(char **xml_data, st_env_t *)	Extracts the service tags from the registry and returns the XML string.
void st_free_xml(char *xml_data)	Frees up memory allocated by st_get_registry_xml
int st_get_registry_doc(xmlDocPtr *doc, st_env_t *)	Extracts the service tags from the registry and returns the document.
int st_get_instance_urns(char *product_urn, char ***instance_urn_pp, st_env_t *)	Returns the count and a pointer to a list of instance URNs for a given product URN.
void st_free_instance_urns(char *[])	Frees up memory allocated by st_get_instance_urns.
char *st_error(int);	Prints a human readable error based on API return code.

The Service Tag structures are as follows:

```
typedef struct st_env_struct {
       char root_dir[PATH_MAX];
       char registry_path[PATH_MAX];
       char registry[PATH_MAX + MAXNAMELEN];
       char registry_urn[MAX_URN_LEN];
       char registry_version[MAX_PRODUCT_VERSION_LEN];
       int  initialized_flag;
} st_env_t;
```

```
typedef struct st_struct {
        char instance_urn[MAX_URN_LEN];
        char product_name[MAX_PRODUCT_NAME_LEN];
        char product_version[MAX_PRODUCT_VERSION_LEN];
        char product_urn[MAX_URN_LEN];
        char product_parent_urn[MAX_URN_LEN];
        char product_parent[MAX_PRODUCT_PARENT_LEN];
        char product_defined_inst_id[MAX_URN_LEN];
        char product_vendor[MAX_PRODUCT_VENDOR_LEN];
        char platform_arch[MAX_PLATFORM_ARCH_LEN];
        char status[MAX_STATUS_LEN];
        char timestamp[MAX_TIMESTAMP_LEN];
        char customer_asset_tag[MAX_CUSTOMER_ASSET_TAG_LEN];
        char group_id[MAX_GROUP_ID_LEN];
        char container[MAX_CONTAINER_LEN];
        char source[MAX_SOURCE_LEN];
        int  installer_uid;
} st_t;


#define MAX_URN_LEN                  256
#define MAX_PRODUCT_NAME_LEN         256
#define MAX_PRODUCT_VERSION_LEN       64
#define MAX_PRODUCT_PARENT_LEN       256
#define MAX_PRODUCT_VENDOR_LEN        64
#define MAX_PLATFORM_ARCH_LEN         64
#define MAX_STATUS_LEN                20
#define MAX_TIMESTAMP_LEN             25
#define MAX_CUSTOMER_ASSET_TAG_LEN  1024
#define MAX_GROUP_ID_LEN              64
#define MAX_CONTAINER_LEN             64
#define MAX_SOURCE_LEN                64
```

10 Appendix F: Service Tag API Error Codes

```
typedef enum {
        ST_ERR_NO_ERROR           = 0,       /* No error */
        ST_ERR_ALREADY_INIT       = -32000,  /* Environment already initialized */
        ST_ERR_NOT_INIT           = -32001,  /* Environment not initialized */
        ST_ERR_GEN_UUID           = -32002,  /* Unable to generate UUID */
        ST_ERR_NO_MEM             = -32005,  /* Unable to allocate memory */
        ST_ERR_NO_DIR             = -32010,  /* Directory or file not found */
        ST_ERR_NOT_AUTH           = -32011,  /* Not authorized */
        ST_ERR_XML_PARSE          = -32020,  /* Unable to parse XML */
        ST_ERR_XML_EMPTY_DOC      = -32021,  /* XML file is empty */
        ST_ERR_XML_NEW_DOC        = -32022,  /* xmlNewDoc failed */
        ST_ERR_XML_NEW_NODE       = -32023,  /* xmlNewNode failed */
        ST_ERR_XML_CREATE_SUB     = -32024,  /* xmlCreateIntSubset failed */
        ST_ERR_XML_SAVE_REG       = -32025,  /* xmlSaveFileEnc failed */
        ST_ERR_XML_INVALID_REG    = -32026,  /* Invalid XML */
        ST_ERR_DUP_REC            = -32030,  /* Duplcate record */
        ST_ERR_REC_NOT_FOUND      = -32031,  /* Record not found */
        ST_ERR_MISSING_FIELD      = -32032,  /* Missing required field(s) */
        ST_ERR_MAX_LEN_EXCEEDED   = -32033,  /* Max. field length exceeded */
        ST_ERR_INVALID_STATUS     = -32034   /* Invalid status code */
} st_error_codes;
```