Sip Advanced Messaging, Usability, Routing and Accounting Interface *

Specification Version 1.10

sipgate GmbH

April 15, 2015

*SAMURAI; The word 'samurai' is derived from the Japanese verb 'sabu', meaning 'to serve'.

This document describes a system for transportation of configuration and auxiliary services data between an end user's CPE and the service provider's SIP system. The CPE may download configuration data for automatic setup for the demanded service as well as setting options on the provider's system to adapt the service to individual needs. Examples for auxiliary services data may be *unified messaging*[1], history listings, account statements, server-side phonebook, etc.

---

[1]Unified Messaging (or UM) is the integration of different streams of messages (email, Fax, voice, video, etc.) into a single in-box, accessible from a variety of different devices. [**?**]

Contents

1. Introduction

1.1. Technical

There are two main groups of target devices the interface is intended to be used by: Customizable devices (like PCs, PDAs, etc) and highly embedded devices (SIP Phones). Communication is done via remote procedure calls (RPC) transported via session layer protocols as e.g. HTTP. Security, such as authentication and encryption, is delegated to those session layer protocols. Transmitted data includes sensitive private data, so a choice of appropriate security mechanisms is highly recommended.

1.2. 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 RFC 2119[**?**].

2. RPC Transport

The service specified by this document may be used via, but is not limited to, the following underlying layers:

2.1. XML-RPC

Implements the *XML-RPC specification*[**?**] where remote procedure calls are encoded in XML[1] and sent in the body of HTTP messages as *text/xml* mimetype. To ensure privacy of user data HTTP traffic is secured by using SSL/TLS[**?**]. Authentication to the service is accomplished by using *HTTP Basic Authentication*[**?**]. The encoding is *UTF-8*[**?**].

Important   The *XML-RPC specification* of *dateTime.iso8601* lacks support for time-zone information. Using *XML-RPC*, all parameters of type *dateTime* (see section 3.1.1 *Data Types*) MUST be declared as *string* in *XML-RPC*. The values of data type *float* used in this specification MUST be declared as *double* in *XML-RPC*. All other data types specified in section 3.1.1 *Data Types* match those defined by *XML-RPC*.

---

[1]Examples of encoding are given in appendix D.

3. RPC Methods

3.1. RPC Method Usage

3.1.1. Data Types

Definition of data types used in this specification. For information about the mapping of these types to those supported by the RPC layer refer to chapter 2 *RPC Transport.*

Type	Description
array	Single- or multidimensional list of data values.
boolean	Boolean value, where 1 = true and 0 = false.
dateTime	Date and time formatted corresponding to RFC 3339[**?**], e.g. "2005-12-30T14:32:57+01:00".
float	Floating-point data with a maximum precision of two decimal places. This specification doesn't provide for exponentional notation.
int	Integer in the range -2147483648 to +2147483647, inclusive.
string	For strings listed in this specification, a maximum allowed length may be listed using the form string(N), where N is the maximum string length in characters and may be "-" to indicate that the size is not limited by this specification. For all strings a maximum length is either explicitly indicated or implied by the size of the elements composing the string. If a string does not have an explicitly indicated maximum length, the default maximum is 256 characters. Strings longer than the specified maximum MAY result in an "parameter exceeds maximum size" error response.

Table 3.1.: Data Types

3.1.2. Other Requirements

All methods must be called using the exact number of arguments specified in this document. Methods called with either missing arguments or extra arguments will

generate an error response. Argument order must be as specified in this document. Future versions of this specification should not redefine the RPC methods defined in this document unless the changes are fully compatible to all prior specifications. Any changes needed in a future version should result in new RPC methods with distinct names being defined.

3.1.3. comments to specification

Argument names may be wrapped to increase readability of tables. In these cases the wraping character "-" is not part of the argument name.

3.2. Mandatory Server Methods

The methods listed in this section MUST be supported by the server. To allow for a maximum of flexibility, only a minimum set of methods is mandatory.

3.2.1. system.listMethods

Request a list of the RPC methods offered by the server with each entry being the name of a method stored as string value.

Argument	Type	Description
none	void	This method has no calling arguments.

Table 3.2.: system.listMethods calling arguments

Argument	Type	Description
listMethods	array of string	List of local method names as strings.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.3.: system.listMethods response arguments

3.2.2. system.methodHelp

Request a human readable description of the given method containing information about purpose and usage.

Argument	Type	Description
MethodName	string	Name of the method (to which help is demanded) as string type value.

Table 3.4.: system.methodHelp calling arguments

Argument	Type	Description
methodHelp	string	Help text or description to the specified mehod.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.5.: system.methodHelp response arguments

3.2.3. system.methodSignature

Request the signature of the given RPC method. The server will return a list of signatures valid for the specified method. The signature is defined as the promised return value's type followed by the types of any arguments. A method MAY be called in more than one way, which is what the purpose of defining signatures is meant to distinguish.

Argument	Type	Description
MethodName	string	Name of the Method (to which a signature is demanded) as string type value.

Table 3.6.: system.methodSignature calling arguments

Argument	Type	Description
methodSignature	array of array of string	List of signatures valid for the method.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.7.: system.methodSignature response arguments

3.2.4. **system.serverInfo**

The *system.serverInfo* method provides specific information on the server software.

Argument	Type	Description
none	void	This method has no calling arguments.

Table 3.8.: system.serverInfo calling arguments

Argument	Type	Description
SpecificationVersion	string	The version number of the implemented SAMURAI specification.
ServerName	string	The name of the server software.
ServerVersion	string	The version number of the server software.
ServerVendor	string	The name of the server software vendor (e.g. company or organization name).
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.9.: system.serverInfo response arguments

3.3. **Optional Server Methods**

The methods listed in this section MAY be supported by server and client so the client MUST discover (eg. see 3.2.1) the methods available on the server after login.

3.3.1. samurai.AccountStatementGet

Request an account statement for a specified period. This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 500, 511

Argument	Type	Description
PeriodStart	dateTime	Start date and time of demanded period.
PeriodEnd	dateTime	End date and time of demanded period.

Table 3.10.: samurai.AccountStatementGet calling arguments

Argument	Type	Description
PeriodStart	dateTime	Start date and time of demanded period.
PeriodEnd	dateTime	End date and time of demanded period.
AccountStatement-ChargedServices	array of struct	List of structures (see table 3.12) listing demanded service and corresponding charges.
BalanceStart	pricetype	Structure as defined in table 3.13, containing the account balance at the start of the given period.
BalanceEnd	pricetype	Structure as defined in table 3.13, containing the account balance at the end of the given period.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.11.: samurai.AccountStatementGet response arguments

Argument	Type	Description
Timestamp	dateTime	Date and time the transaction was made. (This will not be set in cases where uses of a service are grouped.) *optional*
ServiceName	string	e.g. "local call"
Quantity	int	Quantity of taken services of type *ServiceName*.
TotalDuration	int	Total duration of sessions of type *ServiceName* in seconds. (This information may not be available in all cases.) *optional*
Price	pricetype	Structure as defined in table 3.13, containing all pricing information including values in- and excluding tax and tax information.

Table 3.12.: *AccountStatementChargedServices* struct specification

Argument	Type	Description
TotalExcludingVat	float	Balance or total charge excluding VAT for the services demanded during the specified period. (This information may not be available in all cases, but MUST be if *TotalIncludingVat* is not available.) *optional*
TotalIncludingVat	float	Balance or total charge including VAT for the services demanded during the specified period. (This information may not be available in all cases, but MUST be if *TotalExcludingVat* is not available.) *optional*
VatAmount	float	Payable VAT as value in *currency*. (This information may not be available in all cases.) *optional*
VatPercent	float	Tax percentage payable (eg. 16% tax as "16.00").
Currency	string	Currency code as defined by the ISO 4217 standard[**?**] (e.g. "USD" or "EUR").

Table 3.13.: *pricetype* struct specification

3.3.2. samurai.BalanceGet

Request the current balance of the user's account.

Argument	Type	Description
none	void	This method has no calling arguments.

Table 3.14.: samurai.BalanceGet calling arguments

Argument	Type	Description
BalanceTime	dateTime	Date and time the demanded balance is valid at.
CurrentBalance	pricetype	Amount of money in *Currency* that is available at user's account at *BalanceTime* (structure *pricetype* is defined in table 3.13).
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.15.: samurai.BalanceGet response arguments

3.3.3. samurai.ClientIdentify

The client SHOULD identify itself to the server including version and vendor information to allow for better support in cases of problems. It is recommended to call this method once during startup of the client.

Argument	Type	Description
ClientName	string	Name of the client. *optional*
ClientVersion	string	Version of the client. *optional*
ClientVendor	string	Vendor of the client. *optional*

Table 3.16.: samurai.ClientIdentify calling arguments

Argument	Type	Description
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.17.: samurai.ClientIdentify response arguments

3.3.4. samurai.EventDelete

Delete the specified event from the user's account. This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 510, 526

Argument	Type	Description
EventID	string	Unique identifier of the event that is to be deleted.

<div align="center">Table 3.18.: samurai.EventDelete calling arguments</div>

Argument	Type	Description
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

<div align="center">Table 3.19.: samurai.EventDelete response arguments</div>

3.3.5. samurai.EventListGet

Request the list of events. Filter criteria MAY be specified for narrowing down the returned list. If no filter criteria are specified, a complete listing of all events is returned. Filter criteria *Labels*, *EventIDs* and time frame (specified by *PeriodStart* and *PeriodEnd*) will be combined in an AND-condition if specified. *Limit* and *Offset* MAY be specified to allow for pagination of the result list. *IncrementBaseID* MAY be specified to access incremental updates of the user's event list based on the client's most recent event. This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 500, 505, 510, 519, 526

Argument	Type	Description
Labels	array of string	None, one or more labels as filtering argument.
EventIDs	array of string	None, one or more *EventID* as filtering argument. *optional*
TOS	array of string	None, one or more TOS. *optional*
Limit	int	Number of elements the listing is to be limited to. With a value of 0 no limit will be applied.
Offset	int	Number of elements to skip previous to the returned listing. With a value of 0 the result set will start with the first element that matches all filter criteria.
PeriodStart	dateTime	Start date and time for demanded listing. *optional*
PeriodEnd	dateTime	End date and time for demanded listing. *optional*
IncrementBaseID	string	An *EventID* on which an incremental list is to be based. Given the last ID known by the client, the returned listing will only contain consecutive elements. *optional*

Table 3.20.: samurai.EventListGet calling arguments

Argument	Type	Description
EventList	array of struct	Array of none, one or more structures (see table 3.22) listing the stored messages.
OverallCount	int	Number of elements matching the calling arguments, independent of *Offset* and *Limit*.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.21.: samurai.EventListGet response arguments

Argument	Type	Description
EventID	string	Unique identifier assotiated to the event stored on the server.
SourceUri	string	SIP URI of the sender of the event.
TargetUri	string	SIP URI the event was sent to.
Timestamp	dateTime	Date and time the event was received.
SizeDepend	int	Interpretation of the event size value depends on the event's type of service. Types of service are defined in appendix B. This value MAY differ from content specific *SizeDepend* (table 3.23). Note that size data MAY be approximated! *optional*
TOS	string	The event's type of service. Types of service are defined in appendix B.
UnRead	boolean	Stating the "read" / "unread" status of the event.
ContentList	array of struct	List of one or more structures as specified in table 3.23. Note that events MAY lack any downloadable content or may have multiple elements available to download.
Labels	array of string	List of labels assigned to the event.

Table 3.22.: *EventListEntry* struct specification

Argument	Type	Description
SizeBytes	int	Event size measured in Bytes.
SizeDepend	int	Interpretation of the message size value depends on the event's type of service. Types of service are defined in appendix B. Note that size data MAY be approximated! *optional*
DownloadUriHttp	string	Uri to download the current element via HTTP.
ContentType	string	*Media type*[**?**] of the current element.

Table 3.23.: EventContent struct specification

3.3.6. samurai.EventSummaryGet

Request a summary of read and unread events. A list of labels and / or TOS and / or local URIs may be given for filtering. If no parameters are specified, a single summary, including events is returned. This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 519, 526

Argument	Type	Description
LabelName	array of string	None, one or more label names. *optional*
TOS	array of string	None, one or more TOS. *optional*

Table 3.24.: samurai.EventSummaryGet calling arguments

Argument	Type	Description
EventSummary	array of struct	List of one or more structures as specified in table 3.26.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.25.: samurai.EventSummaryGet response arguments

Argument	Type	Description
Read	int	Sum of all messages with status "*read*" that are stored in the user's UM-box and apply to the set filter parameters (see method description).
Unread	int	Sum of all events with status "*unread*" that apply to the set filter parameters (see method description).
LabelName	string	Name of the label read/unread-information is valid for. *optional*
TOS	string	TOS read/unread-information is valid for. *optional*

Table 3.26.: EventSummary struct specification

17

3.3.7. samurai.EventReadSet

Flag one or more events as "was read". This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 510, 526

Argument	*Type*	*Description*
EventIDList	array of string	One or more *EventIDs*.

Table 3.27.: samurai.EventReadSet calling arguments

Argument	*Type*	*Description*
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.28.: samurai.EventReadSet response arguments

3.3.8. samurai.EventUnreadSet

Flag one or more events as "is unread". This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 510, 526

Argument	*Type*	*Description*
EventIDList	array of string	One or more *EventIDs*.

Table 3.29.: samurai.EventUnreadSet calling arguments

Argument	*Type*	*Description*
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.30.: samurai.EventUnreadSet response arguments

3.3.9. samurai.HistoryGetByDate

Request a history listing for a given list of local SIP URI. If no local SIP URI is specified, a listing of all sessions of all local SIP URIs is returned. The listing may also be filtered by given status (e.g. "outgoing" to get a history listing of all outgoing sessions). If no status is specified, the history listing will include sessions of all status types. Standard session status values are "outgoing", "accepted" and "missed". Vendor specific status values MAY be defined but MUST have a "X-" prefix to prevent collisions with future extensions of this specification. Additionally the listing MAY be limited by specifying start and / or end date and time. This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 501, 506, 509

Argument	Type	Description
LocalUriList	array of string	List of local SIP URI that the history listing is demanded for. (May be emtpy for matching all local SIP URIs.) *optional*
StatusList	array of string	List of status that the history listing is to be filtered by (e.g. "outgoing"; may be empty for complete listing). *optional*
PeriodStart	dateTime	Start date and time for demanded listing. *optional*
PeriodEnd	dateTime	End date and time for demanded listing. *optional*

Table 3.31.: samurai.HistoryGetByDate calling arguments

Argument	Type	Description
History	array of struct	List of structures of type *HistoryEntry* (see table 3.33) that each carry information of a session.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.32.: samurai.HistoryGetByDate response arguments

Argument	Type	Description
EntryID	string	Unique identifier for the history entry.
Timestamp	dateTime	Date and Time of session start.
TOS	string	Describing the type of service used. Types of service are defined in appendix B.
LocalUri	string	Local SIP URI of the session.
RemoteUri	string	Remote SIP URI of the session.
Status	string	Describing if the session was either "accepted", "missed" or "outgoing". Vendor specific status values may be defined, but MUST have "X-" as prefix.

Table 3.33.: *HistoryEntry* struct specification

3.3.10. samurai.ItemizedEntriesGet

Request a listing of itemized entries of a specified period of time and specified local SIP URI(s). If no local SIP URI is specified, the listing will include data of all local SIP URIs. This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 500, 501, 506, 511

Argument	Type	Description
LocalUriList	array of string	List of local SIP URI that the listing is demanded for. *optional*
PeriodStart	dateTime	Start date and time of demanded period.
PeriodEnd	dateTime	End date and time of demanded period.

Table 3.34.: samurai.ItemizedEntriesGet calling arguments

Argument	Type	Description
PeriodStart	dateTime	Start date and time of demanded period.
PeriodEnd	dateTime	End date and time of demanded period.
ItemizedEntries	array of struct	List of structures of type *ItemizedEntry* (see table 3.36) that each carry information of a session.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.35.: samurai.ItemizedEntriesGet response arguments

Argument	Type	Description
Timestamp	dateTime	Date and Time of session start.
SourceUri	string	SIP URI the session was originating from.
TargetUri	string	SIP URI the session was made to. *optional*
Price	pricetype	Structure as defined in table 3.13, containing the pricing information on costs incurred including values in- and excluding tax and tax information.
SetupFee	pricetype	Structure as defined in table 3.13, containing pricing information on setup fee including values in- and excluding tax and tax information. (This information may not be available if no setup fee was incurred.) *optional*
PricePerUnit	pricetype	Structure as defined in table 3.13, containing pricing information per unit including values in- and excluding tax and tax information. (This information may not be available in all cases.) *optional*
TicksA	int	Duration of the first unit in seconds. (This information may not be available in all cases.) *optional*
TicksB	int	Duration of units after the first unit in seconds. (This information may not be available in all cases.) *optional*
UnitsCharged	int	Seconds/pages/messages (depending on type of service) charged. (This information may not be available in all cases.) *optional*
TariffName	string	e.g. "local call"
Duration	int	Duration of the session in seconds. (This information may not be available in all cases.) *optional*
TOS	string	Describing the service used. Types of service are defined in appendix B.

Table 3.36.: *ItemizedEntry* struct specification

3.3.11. samurai.LabelAttach

Attach a label to an event. Analogue to directory-folders, events may get labels to allow for structured filing. Unlike with folders, events may be assigned multiple labels (e.g. "work" and "todo" and "important"). This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 510, 517, 526

Argument	Type	Description
LabelName	string	Name of the label that is to be attached to the event identified by *EventID*.
EventID	string	Unique identifier of the event that is to be tagged with *LabelName*.

Table 3.37.: samurai.LabelAttach calling arguments

Argument	Type	Description
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.38.: samurai.LabelAttach response arguments

3.3.12. samurai.LabelDetach

Detach a specified label (cp. 3.3.11 samurai.LabelAttach) from an event. This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 510, 518, 526

Argument	Type	Description
LabelName	string	Name of the label that is to be detached from to the event identified by *EventID*.
EventID	string	Unique identifier of the event from that *LabelName* is to be detached.

Table 3.39.: samurai.LabelDetach calling arguments

Argument	Type	Description
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.40.: samurai.LabelDetach response arguments

3.3.13. samurai.LabelDelete

Completely delete a specified label (cp. 3.3.11 samurai.LabelAttach). Before being deleted, the label will be detached from all appropriate events. Note that some labels may not be deleted (cp. table C.1 in appendix C). This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 519, 526

Argument	Type	Description
LabelName	string	Name of the label that is to be deleted.

Table 3.41.: samurai.LabelDelete calling arguments

Argument	Type	Description
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.42.: samurai.LabelDelete response arguments

3.3.14. samurai.LabelList

Request a listing of all available labels (cp. 3.3.11 samurai.LabelAttach). This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 526

Argument	Type	Description
none	void	This method has no calling arguments.

Table 3.43.: samurai.LabelList calling arguments

24

Argument	Type	Description
LabelList	array of string	List of available labels (may be empty).
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.44.: samurai.LabelList response arguments

3.3.15. samurai.LabelRename

Rename a specified label (cp. 3.3.11 samurai.LabelAttach). Note that some labels may not be modified (cp. table C.1 in appendix C). This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 517, 519, 526

Argument	Type	Description
OldLabel	string	Old name of the label that is to be renamed.
NewLabel	string	New name of the label that is to be renamed.

Table 3.45.: samurai.LabelRename calling arguments

Argument	Type	Description
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.46.: samurai.LabelRename response arguments

3.3.16. samurai.OwnUriListGet

Request the list of SIP URIs belonging to the user's account.

Argument	Type	Description
none	void	This method has no calling arguments.

Table 3.47.: samurai.OwnUriListGet calling arguments

Argument	Type	Description
OwnUriList	array of struct	One or more structures of type *OwnUri* (see table 3.49) that each describe a SIP URI associated to the user's account.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.48.: samurai.OwnUriListGet response arguments

Argument	Type	Description
SipUri	string	SIP URI associated to the user's account.
E164Out	string	*E.164, The international public telecommunication numbering plan* conforming number associated with *SipUri* set for outgoing sessions. (This information may not be available in all cases.) *optional*
E164In	array of string	List of *E.164, The international public telecommunication numbering plan* conforming numbers associated with *SipUri* for incoming sessions. (This information may not be available in all cases.) *optional*
TOS	array of string	List of service types for *SipUri*. Types of service are defined in appendix B.
DefaultUri	boolean	MUST be set once (but only once) per "OwnUriList" and (!) "TypeOfService"; *DefaultUri* will be used for CLIP.
UriAlias	string	Alias name defined for *SipUri*. (This information may not be available in all cases.) *optional*

Table 3.49.: *OwnUri* struct specification

3.3.17. samurai.PhonebookEntryDelete

Delete one or more specified entries from the phonebook stored on the server. This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 510

Argument	Type	Description
EntryIDList	array of string	List of unique identifiers of the phonebook entries to be deleted.

Table 3.50.: samurai.PhonebookEntryDelete calling arguments

Argument	Type	Description
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.51.: samurai.PhonebookEntryDelete response arguments

3.3.18. samurai.PhonebookEntryGet

Request one or more specified entries from the phonebook stored on the server. A hash on the phonebook-entries is provided by the server to allow the client to keep track of changes. Since hash-computation is done serverside, it is up to the server implementor to choose an adequate hash function. This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 510

Argument	Type	Description
EntryIDList	array of string	List of unique identifiers of the phonebook entries that are demanded.

Table 3.52.: samurai.PhonebookEntryGet calling arguments

Argument	Type	Description
EntryList	array of struct	One or more structures of type *PhonebookEntry* (see table 3.54) that each describe a phonebook entry stored on the server.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.53.: samurai.PhonebookEntryGet response arguments

Argument	Type	Description
EntryID	string	Unique identifier of the phonebook entry that is demanded.
EntryHash	string	Hashvalue on the phonebookentry (cp. *3.3.18 samurai.PhonebookEntryGet*).
Entry	string(-)	"vCard - The Electronic Business Card - Version 2.1"[**?**]. Note that this document does not specify a maximum size of the string containing vCARD data.

Table 3.54.: *PhonebookEntry* struct specification

3.3.19. samurai.PhonebookEntrySet

Store a phonebook entry to the serverside phonebook. If the specified EntryID already exists in the phonebook, that entry will be replaced with the new one. If EntryID is left blank or there is no existing entry that matches the given EntryID, the entry will be added to the phonebook as new. This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 510, 514, 520

Argument	Type	Description
EntryList	array of struct	List of structures of type *PhonebookEntryIn-Struct* (see table 3.56) that each provides ID (*optional*) and "vCard - The Electronic Business Card - Version 2.1"[**?**] of one phonebook entry.

Table 3.55.: samurai.PhonebookEntrySet calling arguments

Argument	Type	Description
EntryID	string	Unique identifier of the phonebook entry that is to be written.
Entry	string(-)	"vCard - The Electronic Business Card - Version 2.1"[**?**]. Note that this document does not specify a maximum size of the string containing vCARD data.

Table 3.56.: *PhonebookEntrySetInStruct* struct specification

Argument	Type	Description
EntryList	array of struct	List of structures of type *PhonebookEntrySetOutStruct* (see table 3.58) that each provides ID and hashvalue of one phonebook entry.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.57.: samurai.PhonebookEntrySet response arguments

Argument	Type	Description
EntryID	string	Unique identifier of the phonebook entry that was written.
EntryHash	string	Hashvalue on the phonebookentry (cp. *3.3.18 samurai.PhonebookEntryGet*).

Table 3.58.: *PhonebookEntrySetOutStruct* struct specification

3.3.20. samurai.PhonebookListGet

Request the list of all EntryIDs and corresponding hash-values of the serverside phonebook.

Argument	Type	Description
none	void	This method has no calling arguments.

Table 3.59.: samurai.PhonebookListGet calling arguments

Argument	Type	Description
PhonebookList	array of struct	List of structures of type *PhonebookListEntry* (see table 3.61) that each stores ID and hash-value of one phonebook entry.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.60.: samurai.PhonebookListGet response arguments

Argument	Type	Description
EntryID	string	Unique identifier referring to an entry of the user's phonebook on the server.
EntryHash	string	Hashvalue on the phonebookentry (cp. *3.3.18 samurai.PhonebookEntryGet*).

Table 3.61.: *PhonebookListEntry* struct specification

3.3.21. samurai.RecommendedIntervalGet

Request the time in seconds the service provider recommends as minimum interval for calling a specific method. One or more method names may be specified in a list. The client SHOULD make use of this method because calling a method more often than recommended MAY cause the server to block the request and return statuscode 412 (cp. table A.2 in appendix A).

Argument	Type	Description
MethodList	array of string	List of method names to request the minimum recommended calling interval for.

Table 3.62.: samurai.RecommendedIntervalGet calling arguments

Argument	Type	Description
IntervalList	array of struct	Structure as defined in table 3.64, containing method name and recommended minimum interval.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.63.: samurai.RecommendedIntervalGet response arguments

Argument	Type	Description
MethodName	string	Name of server method.
RecommendedInterval	int	Minimum interval in seconds the provider recommends for calling method *MethodName*.

Table 3.64.: *RecommendedInterval* struct specification

3.3.22. samurai.ServerdataGet

Request serverdata required for automatical setup of a client for SIP communication including SIP registrar server, SIP outbound proxy, STUN[1] server, NTP[2] server, HTTP server, SAMURAI server, SIMPLE[3] server and XMPP[4] server. Fields that do not apply to the service provider's system MAY be left blank.

Argument	Type	Description
none	void	This method has no calling arguments.

Table 3.65.: samurai.ServerdataGet calling arguments

---

[1]Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs); RFC 3489[?]

[2]Network Time Protocol

[3]Sip for Instant Messaging and Presence Leveraging Extensions (IETF working group)

[4]Extensible Messaging and Presence Protocol; see http://www.xmpp.org/

31

Argument	Type	Description
SipRegistrar	string	Hostname or IP of service provider's SIP registrar server.
SipOutboundProxy	string	Hostname or IP of service provider's SIP outbound proxy server.
StunServer	string	Hostname or IP of service provider's STUN server.
NtpServer	string	Hostname or IP of service provider's NTP server.
HttpServer	string	Hostname or IP of service provider's HTTP server.
SamuraiServer	string	Hostname or IP of service provider's SAMURAI server.
SimpleServer	string	Hostname or IP of service provider's SIMPLE server.
XmppServer	string	Hostname or IP of service provider's XMPP server.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.66.: samurai.ServerdataGet response arguments

3.3.23. samurai.SessionClose

Close a specified session created by *samurai.SessionInitiate* (see 3.3.24). This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 510, 512

Argument	Type	Description
SessionID	string	Unique identifier associated to the session that is to be closed.

Table 3.67.: samurai.SessionClose calling arguments

Argument	Type	Description
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.68.: samurai.SessionClose response arguments

3.3.24. samurai.SessionInitiate

Setup a connection between the user's own sip-client (e.g. IP-Phone) and a given remote SIP URI; commonly known as "click2dial". This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 501, 502, 506, 520, 525

Argument	Type	Description
LocalUri	string	Local SIP URI that the session originates from.
RemoteUri	string	Remote SIP URI that is to be called.
TOS	string	Type of service for the demanded session. Types of service are defined in appendix B.
Content	string(-)	Content to be transmitted. This field only applies to specific types of content, e.g. "text" and is empty otherwise (e.g. for "voice").
Schedule	dateTime	Date and time the session is to be initiated. If *Schedule* is not set the session is initiated immediately. *optional*

Table 3.69.: samurai.SessionInitiate calling arguments

Argument	Type	Description
SessionID	string	Unique identifier associated to the created session.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.70.: samurai.SessionInitiate response arguments

33

3.3.25. samurai.SessionInitiateMulti

Setup a connection between the user's own sip-client (e.g. IP-Phone) and a list of remote SIP URI. This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 501, 502, 506, 520, 525

Argument	Type	Description
LocalUri	string	Local SIP URI that the session originates from.
RemoteUri	array of string	List of remote SIP URI that sessions are to be established to.
TOS	string	Type of service for the demanded sessions. Types of service are defined in appendix B.
Content	string(-)	Content to be transmitted. This field only applies to specific types of content, e.g. "text" and is empty otherwise (e.g. for "voice").
Schedule	dateTime	Date and time the sessions are to be initiated. If *Schedule* is not set the sessions are initiated immediately. *optional*

Table 3.71.: samurai.SessionInitiateMulti calling arguments

Argument	Type	Description
SessionList	array of struct	Array of structures (see table 3.73) listing the IDs linked to the corresponding remote URI.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.72.: samurai.SessionInitiateMulti response arguments

Argument	Type	Description
SessionID	string	Unique identifier associated to the created session.
RemoteUri	string	Remote SIP URI of the initiated session.

Table 3.73.: *SessionListEntry* struct specification

3.3.26. samurai.SessionStatusGet

Request status information for a specified session created by *samurai.SessionInitiate* (see 3.3.24). This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 510, 512

Argument	Type	Description
SessionID	string	Unique identifier associated to the session that status information is requested for.

Table 3.74.: samurai.SessionStatusGet calling arguments

Argument	Type	Description
SessionStatus	string	Human readable status information for the referenced session.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.75.: samurai.SessionStatusGet response arguments

3.3.27. samurai.TosListGet

Request a listing of all available types of service (TOS). TOS are defined in appendix B.

Argument	Type	Description
none	void	This method has no calling arguments.

Table 3.76.: samurai.TosListGet calling arguments

Argument	Type	Description
TosList	array of string	List of available TOS.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.77.: samurai.TosListGet response arguments

3.3.28. samurai.UmSummaryGet

This method is obsolete[5], use 3.3.6 *samurai.EventSummaryGet* instead!

Request a summary of read and unread messages stored in the user's unified messaging (UM) box. A list of labels and / or TOS and / or local URIs may be given for filtering. If no parameters are specified, a single summary, including all messages stored in the user's UM box is returned. This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 519, 526

Argument	Type	Description
LabelName	array of string	None, one or more label names. *optional*
TOS	array of string	None, one or more TOS. *optional*
LocalUriList	array of string	None, one or more local URIs. *optional*

Table 3.78.: samurai.UmSummaryGet calling arguments

Argument	Type	Description
UmSummary	array of struct	List of one or more structures as specified in table 3.80.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.79.: samurai.UmSummaryGet response arguments

---

[5]Though still supported.

Argument	Type	Description
Read	int	Sum of all messages with status "*read*" that are stored in the user's UM-box and apply to the set filter parameters (see method description).
Unread	int	Sum of all messages with status "*unread*" that are stored in the user's UM-box and apply to the set filter parameters (see method description).
LabelName	string	Name of the label read/unread-information is valid for. *optional*
TOS	string	TOS read/unread-information is valid for. *optional*

Table 3.80.: UmSummary struct specification

3.3.29. samurai.UserdataGreetingGet

Request data from the server to be used for personalized greetings comprehending gender, first and last name.

Argument	Type	Description
none	void	This method has no calling arguments.

Table 3.81.: samurai.UserdataGreetingGet calling arguments

Argument	Type	Description
Gender	string	Gender stored in the user's account.
FirstName	string	First name stored in the user's account.
LastName	string	Last name stored in the user's account.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.82.: samurai.UserdataGreetingGet response arguments

3.3.30. samurai.UserdataSipGet

Request the SIP userdata for authentication to the SIP server. This method MAY return the following *method specific server status codes* (cp. table A.2 in appendix A): 501, 506, 527

Argument	Type	Description
LocalUriList	array of string	List of local SIP URI that SIP userdata is demanded for.

Table 3.83.: samurai.UserdataSipGet calling arguments

Argument	Type	Description
SipDataList	array of struct	List of one or more structures as specified in table 3.85.
StatusCode	int	Numeric status code as defined in appendix A.
StatusString	string	Human readable description of *StatusCode*.

Table 3.84.: samurai.UserdataSipGet response arguments

Argument	Type	Description
LocalUri	string	Local SIP URI that SIP userdata is demanded for.
SipUserID	string	Unique identifier for authentication to the SIP-server.
SipPassword	string	Authentication password for authentication to the SIP-server.

Table 3.85.: *UserdataSip* struct specification

3.4. Vendor Specific Extensions

Vendor specific methods MAY be added, but MUST have "X-" as name prefix to prevent collisions with future extensions of this specification.

A. Server Status Codes

The *general server status codes* listed in table A.1 may be returned by any method called and thus MUST be understood by any client. *Method specific server status codes* shown in table A.2 MUST be understood by any client that implements the appropriate methods. Applying *method specific server status codes* are referenced to in the respective method descriptions above.

Status Code	Description
200	Method success
400	Method not supported
401	Request denied (no reason specified)
402	Internal error
403	Invalid arguments
404	Resources exceeded (this MUST not be used to indicate parameters in error)
405	Invalid parameter name
406	Invalid parameter type
407	Invalid parameter value
408	Attempt to set a non-writable parameter
409	Notification request rejected.
410	Parameter exceeds maximum size.
411	Missing parameter.
412	Too many requests.

Table A.1.: general server status codes

Status Code	Description
500	Date out of range.
501	Uri does not belong to user.
502	Unknown type of service.
503	Selected payment method failed.
504	Selected currency not supported.
505	Amount exceeds limit.
506	Malformed SIP URI.
507	URI not in list.
508	Format is not valid E.164.
509	Unknown status.
510	Unknown ID.
511	Invalid timevalue.
512	Referenced session not found.
513	Only single default per TOS allowed.
514	Malformed VCARD format.
515	Malformed PID format.
516	Presence information not available.
517	Invalid label name.
518	Label not assigned.
519	Label doesn't exist.
520	Parameter includes invalid characters.
521	Bad password. (Rejected due to security concerns.)
522	Malformed timezone format.
523	Delay exceeds limit.
524	Requested VPN type not available.
525	Requested TOS not available.
526	Unified messaging not available.
527	URI not available for registration.

Table A.2.: method specific server status codes

Status Code	Description
900 - 999	Vendor defined status codes

Table A.3.: server status codes of vendor specific extensions

B. TOS (type of service) specification

The provider may offer different services as for example voice- and video-communication
to it's customers. The different services demand for different treatment in terms of
configuration and billing, as well as in terms of presentation to the customer. To take
these constraints into account, a *type of service* (TOS) identifier is defined as table
B.1 shows. Additional to specifying the types of service, the measuring unit for mes-
sages of each type is defined, e.g. the "interpreted size" of a fax-message is measured
in "pages" and the "interpreted size" of a voice-message is measured in "seconds".
Vendor specific types of service may be used, but MUST have "X-" as the identifier's
prefix as well as a defined measuring unit for belonging messages.

identifier	*units*	*comment*
fax	pages	fax transmission
text	characters	text message (e.g. "SMS")
video	seconds	video communication
voice	seconds	voicemail
X-call	seconds	voice call

Table B.1.: *type of service* specification

C. system labels

Analogue to directory-folders, events may get labels to allow for structured filing. Unlike with folders, events may be assigned multiple labels (e.g. "work" and "todo" and "important"). Although label names can be chosen by the user some names are reserved for system use. These reserved *system labels* are listed in table C.1.

label name	description
inbox	incoming events
sent	outgoing events
spam	events categorized as "unwanted"
pending	events pending to be sent
sending	events that are currently being sent
trash	trashed events
failed	failed actions on events will result in this label being attached
scheduled	events scheduled for later sending
archive	"archived" events will not be shown in listings based on other label names unless explicitly requested

Table C.1.: reserved label names

D. XMLRPC message examples

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
  <methodName>samurai.AccountStatementGet</methodName>
  <params>
    <param>
      <value><struct>
        <member>
          <name>PeriodEnd</name>
          <value><string>
            2007-07-31T00:00:00
          </string></value>
        </member>
        <member>
          <name>PeriodStart</name>
          <value><string>
            2007-06-01T00:00:00
          </string></value>
        </member>
      </struct></value>
    </param>
  </params>
</methodCall>
```

Listing D.1: XMLRPC method call to *samurai.AccountStatementGet*

```xml
<?xml version="1.0"?>
<methodResponse>
  <params><param>
    <value><struct>
      <member><name>StatusCode</name><value><i4>200</i4></value></member>
      <member>
        <name>PeriodEnd</name><value><string>2007-07-31T00:00:00+0200</string></value>
      </member>
      <member>
        <name>BalanceEnd</name>
        <value><struct>
          <member><name>Currency</name><value><string>EUR</string></value></member>
          <member><name>TotalIncludingVat</name><value><double>2.7204</double></value></member>
        </struct></value>
      </member>
      <member>
        <name>AccountStatementChargedServices</name>
        <value><array><data>
          <value><struct>
            <member>
              <name>ServiceName</name><value><string>Festnetz Deutschland</string></value>
            </member>
            <member>
              <name>Price</name>
              <value><struct>
                <member>
                  <name>Currency</name><value><string>EUR</string></value>
                </member>
                <member>
                  <name>TotalIncludingVat</name><value><double>0.0179</double></value>
                </member>
                <member>
                  <name>VatPercent</name><value><double>0.19</double></value>
                </member>
              </struct></value>
            </member>
            <member><name>Quantity</name><value><i4>1</i4></value></member>
          </struct></value>
          <value><struct>
            <member>
              <name>ServiceName</name><value><string>germany 01801</string></value>
            </member>
            <member>
              <name>Price</name>
              <value><struct>
                <member>
                  <name>Currency</name><value><string>EUR</string></value>
                </member>
                <member>
                  <name>TotalIncludingVat</name><value><double>0.0390</double></value>
                </member>
                <member>
                  <name>VatPercent</name><value><double>0.19</double></value>
                </member>
              </struct></value>
            </member>
            <member><name>Quantity</name><value><i4>1</i4></value></member>
          </struct></value>
        </data></array></value>
      </member>
      <member>
        <name>PeriodStart</name><value><string>2007-06-01T00:00:00+0200</string></value>
      </member>
      <member>
        <name>StatusString</name><value><string>Method success</string></value>
      </member>
      <member>
        <name>BalanceStart</name>
        <value><struct>
          <member><name>Currency</name><value><string>EUR</string></value></member>
          <member><name>TotalIncludingVat</name><value><double>2.7773</double></value></member>
        </struct></value>
      </member>
    </struct></value>
  </param></params>
</methodResponse>
```

Listing D.2: XMLRPC method response of *samurai.AccountStatementGet*

E. SAMURAI specification change log

Version 1.06 released August, 2007

- *official release*

Version 1.07 released July, 2008

- formal release due to corporate name change

Version 1.08 released `May`, 2009

- new appendix C *system labels*
- new method *samurai.PhonebookEntryDelete*
- new method *samurai.PhonebookEntrySet*
- new method *samurai.LabelAttach*
- new method *samurai.LabelDetach*
- new method *samurai.LabelDelete*
- new method *samurai.LabelList*
- new method *samurai.LabelRename*
- new method *samurai.EventDelete*
- new method *samurai.EventListGet*
- new method *samurai.EventSummaryGet*
- new method *samurai.EventReadSet*
- new method *samurai.EventUnreadSet*
- method *samurai.UmSummaryGet* marked obsolete

Version 1.09 released `June`, 2009

- fixed typos in description of method *samurai.EventListGet*
- new parameter *OverallCount* in *samurai.EventListGet* result
- changed section title *HTTPS* to *XML-RPC*
- added note on mapping specification data types to RPC layer data types
- added note about encoding with *XML-RPC*

Version 1.10 released `April`, 2015

- added TOS specification for X-call