Sinch Contact Pro

Presence Synchronization Interface (PSI)

April 2023

Date	Changes
14.04.2022	Product name and template changed
04.01.2018	The document template changed
02.05.2014	BCM changed to SAP Contact Center
22.05.2012	Minor changes
02.05.2012	Initial version

Contents

1 Introduction

This document describes Presence Synchronization Interface (PSI) of the Sinch Contact Pro (previously SAP Contact Center) software. This information is directed to system integrators and third-party software vendors who wish to browse and configure users of a Sinch Contact Pro system.

For information about the general integration interface features, such as authentication, authorization, and audit log use, see Master Guide. Some details of the interface and this document may be changed without prior notice, but the basic principles of the interface (such as the use of SOAP over HTTP) are not subject to change.

This interface is a SOAP interface and is based on the SOAP version 1.1 (http://www.w3.org/TR/SOAP). The interface uses HTTP protocol to carry SOAP messages between a SOAP client and itself. If an operation fails for some reason, a standard SOAP 1.1 fault message is returned.

In the current version of the interface, only the faultcode and faultstring elements are used. The possible fault codes are only Client or Server without any further classification of the error. However, the faultstring element is used for returning a comprehensible description of the error situation.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <soap:Fault>
         <faultcode>soap:Client</faultcode>
         <faultstring>Fault description</faultstring>
         <detail />
      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

2 Architecture Overview

The Presence Synchronization Interface (PSI) provides methods to monitor and change the presence state of Sinch Contact Pro users. Presence state here includes user login status, presence or absence profiles, call state and service state. The interface is meant to be used for synchronizing the user states between Sinch Contact Pro and an external system (for example Microsoft Lync).

PSI uses sessions to separate different entities using the interface. Application using PSI first creates a session which is used as a key to subsequent operations. Application then subscribes the users whose presence it is going to monitor. Presence notifications are sent to each session separately, and they are based on subscriptions created under the session. Sessions time out and are cleared automatically if no calls for a session are received in two minutes; the normal operation for an application using PSI is to continuously call the GetPresenceChanges method to receive the change notifications.

Access to user presence is based on user groups. Each group has a Presence Synchronization Token (later PSToken), and PSI provides access to users belonging to these groups. Additionally, each user in these groups has attribute called Presence Synchronization ID (later PSId) which can be used to bind Sinch Contact Pro user to corresponding user entity in the external system.

Access rights for PSI are set on service level meaning that the authenticated account under which PSI is being used needs to have rights to use the PSI service.

The following methods are provided by the PSI:

- `CreateSession`
- `EndSession`
- `GetUsers`
- `GetPresenceProfiles`
- `SubscribeUserPresence`
- `UnsubscribeUserPresence`
- `GetPresenceChanges`
- `SetUserPresence`
- `SubscribeConfigChanges`
- `GetConfigurationChanges`

Figure 1 shows a principal overview regarding the PSI architecture.

Chapter Interface Specification provides a detailed interface description. Chapter PSI Use Case Scenarios gives examples of the way PSI can be used.

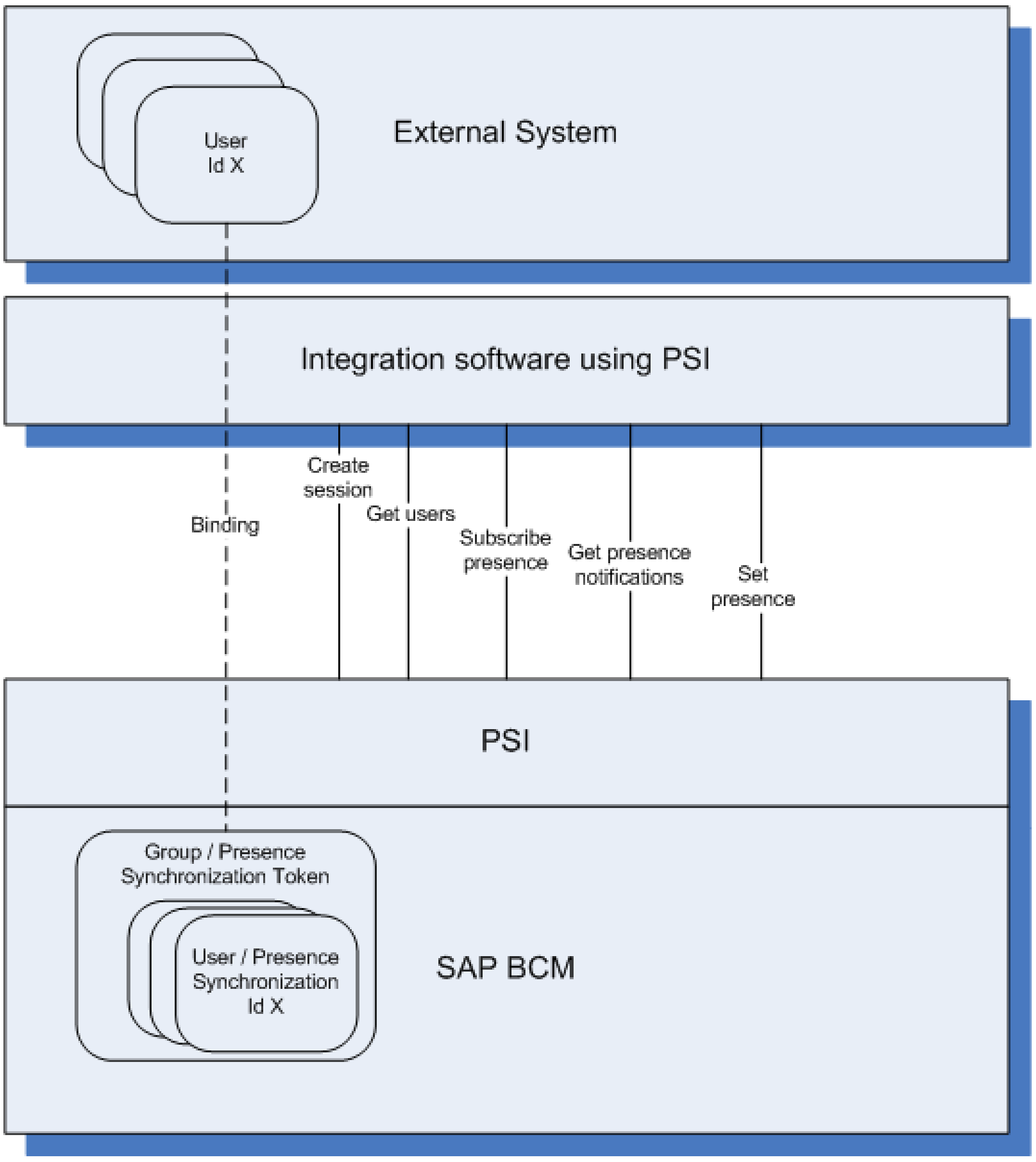
User
Id X
External System
Integration software using PSI
Binding
Create session
Get users
Subscribe presence
Get presence notifications
Set presence
PSI
Group / Presence Synchronization Token
User / Presence Synchronization Id X
SAP BCM

Figure 1: PSI Overview

3 Installing, Configuring, and Using PSI

PSI is installed into a Sinch Contact Pro system as part of the Integration Interfaces software package. For details about the installation procedure and about configuring the interface, see *Installation Guide*.

To use PSI, your application should read the WSDL document of PSI by getting it from the following URL:

```
http://<integration_interfaces_address>:<port>/PSI/Service.asmx?wsdl
```

Where `<integration_interfaces_address>` is replaced with the IP address of the virtual unit where the Integration Interfaces software package is installed, and <port> is replaced with the port which is specified during the installation.

4 Complex Element Description

This section describes the content of complex XML elements that are used in PSI. PSI uses only a very limited set of elements for all input and output parameters.

4.1 Clarification Regarding GUIDs

GUID stands for Globally Unique Identifier. In other words, GUIDs are just ordinary identifiers (IDs). The GUIDs are represented by strings.

The following requirements must be applied when defining valid input GUIDs:

- A GUID may only contain hexadecimal characters or dashes. Uppercase as well as lowercase characters are accepted.
- A GUID may be a string of 32 hex digits with 4 dashes (xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx).
- A GUID may be a string of 32 hex digits without dashes (xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx).

PSI returns GUIDs only in the following format: Output GUIDs are 32 uppercase hexadecimal characters without dashes (xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

4.2 UserType Element

The `UserType` element contains minimal set of properties describing a Sinch Contact Pro user. The `GetUsers` method returns the `UserType` element for each user whose presence is to be synchronized via PSI. Users need to belong to a group with an assigned PSToken attribute and they need to have the PSId attribute for binding.

```
structure UserType

   GUID    BcmID;
   String  ExtID;
   String  PSToken;
   String  FirstName;
   String  SurName;
   String  Language;
   Byte    Type;
```

Element name	UserType
Purpose	Describes a user for whom presence synchronization is applied
Sub-elements	string < BcmID > User GUID, identifies the user in Sinch Contact Pro side. string < ExtID > Presence Synchronization ID, meant for identifying the user in the external system. string < PSToken > Presence Synchronization Token of the group containing the user string < FirstName > User's first name as defined in Sinch Contact Pro – only for informative use string < SurName > User's last name as defined in Sinch Contact Pro – only for informative use string < Language > User's language as defined in Sinch Contact Pro – meant for possible localization in the external system bool < Type > User type – currently unused and meant for possible future extensions

4.3 PresenceProfileType Element

The `PresenceProfileType` element briefly describes a presence profile in the Sinch Contact Pro system. Presence profiles are returned as the result of the `GetPresenceProfiles` call. The GUID member identifies the profile, the `ProfileName` member is given in the Sinch Contact Pro default language and is meant for informative use only (it may not be unique).

```
structure PresenceProfileType

   GUID         ProfileID
   String       ProfileName
```

Element name	PresenseProfileType
Purpose	Defines a presence profile in Sinch Contact Pro
Sub-elements	string < ProfileID > The GUID of the profile in Sinch Contact Pro, identifies the profile. string < ProfileName > Name of the profile in system default language.

4.4 PresenceChangeNotification Element

The `PresenceChangeNotification` element describes the changed/current presence state for a user. For each monitored (subscribed) user, the element is returned in the `GetPresenceChanges` method result whenever the presence state is changed.

```
structure PresenceChangeNotification

     GUID               UserId
     GUID               PresenceProfile
     DateTime           EndTime
     UserLoginState     LoginState
     UserCallState      CallState
     UserServiceState   ServiceState
```

Element name	PresenseChangeNotification
Purpose	The element contains the changed (current) presence state of a single Sinch Contact Pro user.
Sub-elements	string < UserId > User GUID in Sinch Contact Pro, identifying the user string < PresenceProfile > GUID of the current presence profile. DateTime < EndTime > Ending time of the current profile. If the profile is continuous, DateTime:max is used. UserLoginState < LoginState > Enumeration specifying the current logon status (logged in/logged out/no change) UserCallState < CallState > Enumeration specifying the current call state (talking/idle/no change) UserServiceState < ServiceState > Enumeration specifying the current service state (Service/Paperwork/WrapUp/ No change)

4.5 PresenceType Element

The `PresenceType` element describes the presence state for a user. This element is used in the `SetUserPresence` operation when the user's presence is to be changed via PSI.

```
  structure PresenceType

     GUID               PresenceProfile
     DateTime           EndTime
     UserCallState      CallState
     UserServiceState   ServiceState
```

Element name	PresenseType
Purpose	The element contains the presence state for a Sinch Contact Pro user.
Sub-elements	string < PresenceProfile > GUID of the current presence profile. DateTime < EndTime > Ending time of the current profile. If the profile is continuous, DateTime:max is used. UserCallState < CallState > Enumeration specifying the current call state (talking/idle/no change) UserServiceState < ServiceState > Enumeration specifying the current service state (Service/Paperwork/WrapUp/No change)

4.6 ConfigurationChange Element

The `ConfigurationChange` element contains information about configuration changes in the Sinch Contact Pro database related to the PSI use. Change reporting offers a way to reflect changes for applications using PSI: For example, when a new user is created in Sinch Contact Pro, the application using PSI should also start presence monitoring for the new user. Change elements are returned with the `GetConfigurationChanges` method.

```
   Structure ConfigurationChange

      String  ObjectId
      String  ObjectType;
      String  ChangeId;
```

Element name	ConfigurationChange
Purpose	The element represents a change in Sinch Contact Pro configuration related to PSI use.
Sub-elements	string < ObjectId > The identifier of the changed object. Depending on object in question, this ID can be User GUID, Profile GUID or Group PSToken. string < ObjectType > Identifies the changed object. Following types are reported: USER – user change PSTOKEN - group change PROFILE – presence profile change USER_PSID – user presence synchronization id has changed string < ChangeId> String specifying the change type, one of the following values: CREATED – new object has been created DELETED – object has been removed

	MODIFIED – object has been modified

4.7 Enumerations

The following enumerations are used in PSI:

The `EventFlags` enumeration is used in `SubscribeUserPresence` to specify the types of presence changes PSI should report for the user.

The `UserLoginState`, `UserServiceState`, and `UserCallState` enumerations are used in `GetPresenceChanges` to inform the type of change. The value `No_Change` indicates that no change to the previous value has occurred.

```
enum EventFlags
   {
      ProfileChangesOnly,
      Call,
      Login,
      Call_and_Login,
      Service,
      Call_and_Service,
      Login_and_Service,
      All = 7
   }

ProfilesChangesOnly  - Presence profile changes
Call                 - Call state
Login                - Login state
Call_and_Login       - Call and login states
Service              - Agent service state
Call_and_Service     - Call and service states
Login_and_Service    - Login and service states
All                  - All states (profile,login,call,service)

Note that presence profile changes are always reported

enum UserLoginState
   {
      No_Change,
      Logged_Out ,
      Logged_In
   }

No_Change     - Login state has not changed
Logged_Out    - User has logged out
Logged_In     - User has logged in

enum UserServiceState
   {
      No_Change,
      Working,
      Paperwork,
      Wrapup
   }
No_Change  - Service state has not changed
Working    - Service state is set to ‘Working’
Paperwork  - Service state is set to ‘Paperwork’
Wrapup     - Service state is set to ‘WrapUp’ (aka LateAdmin)


enum UserCallState
   {
      No_Change,
      Idle,
      Talking
   }

No_Change  - Call state has not changed
Idle       - No call is active
Talking    - User is talking in SAP Contact Center
```

5 Interface Specification

The Presence Synchronization Interface (PSI) implements the operations listed below. Each operation consists of a message pair (request/response).

Operation	Overview
CreateSession	Creates a session for PSI
EndSession	Ends the PSI session
GetUsers	Get users for presence synchronization
GetPresenceProfiles	Get presence profiles
SubscribeUserPresence	Subscribe to presence change notifications for a user
UnsubscribeUserPresence	Unsubscribe (end subscription) for a user
GetPresenceChanges	Get change indications for the subscribed users
SetUserPresence	Set presence for a user
SubscribeConfigChanges	Subscribe configuration changes for specified groups
GetConfigurationChanges	Get relevant configuration changes (by subscription / all)

5.1 CreateSession Operation

5.1.1 Definition

Purpose:	The CreateSession method creates a session between the PSI service and the SOAP client.
Request name:	CreateSession Sessions time out after two minutes of idling. In other words, PSI invalidates the given session ID if it does not get any session-related requests for two minutes. PSI can currently handle the maximum amount of 32 parallel sessions.
Response name:	CreateSessionResult
Response elements:	The operation returns CreateSessionResult, which contains the ID (GUID) for the new session. This ID is to be used in all subsequent calls related to this session.

5.1.2 CreateSession Message

The `CreateSession` message carries a `CreateSession` operation request from a SOAP client to PSI.

The `CreateSession` message example:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
<soap:Body>
 <CreateSession xmlns="http://sap.com/bcm/PSI" />
</soap:Body>
</soap:Envelope>
```

5.1.3 CreateSessionResponse Message

The `CreateSessionResponse` message carries results of the `CreateSession` operation from PSI back to a SOAP client.

The `CreateSessionResponse` message example:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
 xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
 xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<CreateSessionResponse xmlns=http://sap.com/bcm/PSI >
<CreateSessionResult>
<SessionID>F4DE7406EDA84EA7BDF670E62213FBB4</SessionID>
</CreateSessionResult>
</CreateSessionResponse>
</soap:Body>
</soap:Envelope>
```

5.2 GetPresenceProfiles Message

5.2.1 Definition

Purpose:	The GetPresenceProfiles message returns information of all presence profiles defined in the Sinch Contact Pro database.
Request name:	GetPresenceProfiles
Response name:	GetPresenceProfilesResponse
Response elements:	The operation returns GetPresenceProfilesResult, which is an array of the PresenceProfileType structures. The structure is described here.

5.2.2 GetPresenceProfiles Message

The following is an example of the `GetPresenceProfiles` message.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<GetPresenceProfiles xmlns="http://sap.com/bcm/PSI" />
</soap:Body>
</soap:Envelope>
```

5.2.3 GetPresenceProfilesResponse Message

The `GetPresenceProfilesResponse` message carries the defined profiles to the caller as an array of the `PresenceProfileType` structures.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<GetPresenceProfilesResponse xmlns="http://sap.com/bcm/PSI">
<GetPresenceProfilesResult>
 <PresenceProfile>
  <ProfileID>02770429F80BDF1191AD001CC4D98816</ProfileID>
  <ProfileName>Available</ProfileName>
 </PresenceProfile>
 <PresenceProfile>
  <ProfileID>04770429F80BDF1191AD001CC4D98816</ProfileID>
  <ProfileName>Lunch</ProfileName>
 </PresenceProfile>
 <PresenceProfile>
  <ProfileID>F5A73588445244D4867A5B3A95DDAB48</ProfileID>
  <ProfileName>Vacation</ProfileName>
 </PresenceProfile>
</GetPresenceProfilesResult>
</GetPresenceProfilesResponse>
</soap:Body>
</soap:Envelope>
```

5.3 GetUsers Operation

5.3.1 Definition

Purpose:	The GetUsers method is used to retrieve information about Sinch Contact Pro users belonging to groups identified by the given Presence Synchronization Tokens.	
Request name:	GetUsers	
Request elements:	Session ID	Specifies the session ID retrieved from the CreateSession call.
	Array of Presence Synchronization Tokens <PSToken>	Specifies the target Sinch Contact Pro users the requesting user is interested in. The given PSToken identifies the group in Sinch Contact Pro: users belonging to a group that has the same Presence Synchronization Token are returned in the result. One or several PSTokens can be provided. The maximum number of PSTokens per one GetUsers call is currently 64.
Response name:	GetUsersResponse	
Response elements:	The operation returns GetUsersResult, which is an array of user elements as described in chapter ConfigurationChange Element.	

5.3.2 GetUsers Message

The `GetUsers` message carries a `GetUsers` operation request and its parameters from a SOAP client to PSI.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<GetUsers xmlns="http://sap.com/bcm/PSI">
<sessionId>F4DE7406EDA84EA7BDF670E62213FBB4</sessionId>
<PSToken><string>basic</string></PSToken>
</GetUsers>
</soap:Body>
</soap:Envelope>
```

5.3.3 GetUsersResponse Message

The `GetUsersResponse` message carries results of the `GetUsers` operation from PSI back to a SOAP client.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<GetUsersResponse xmlns="http://sap.com/bcm/PSI">
<GetUsersResult>
<User>
<BcmID>648E588B1B0C439289627AF79869CFC2</BcmID>
<ExtID>psid_for_test1</ExtID>
<PSToken>basic</PSToken>
<FirstName>Test1</FirstName>
<SurName>User</SurName>
<Language>EN</Language>
<Type>0</Type>
</User>
<User><BcmID>5F96370796274E7391FEA1F778455159</BcmID>
<ExtID>psid_for_test2</ExtID>
<PSToken>basic</PSToken>
<FirstName>Test2</FirstName>
<SurName>User</SurName>
<Language>EN</Language>
<Type>0</Type>
</User>
</GetUsersResult>
</GetUsersResponse>
</soap:Body>
</soap:Envelope>
```

5.4 SubscribeUserPresence Operation

5.4.1 Definition

Purpose:	The SubscribeUserPresence operation is used to subscribe the presence change notifications for a Sinch Contact Pro user.	
Request name:	SubscribeUserPresence	
Request elements:	SessionID <SessionID>	Specifies the session ID retrieved from the CreateSession call.
	UserId <UserId>	Sinch Contact Pro ID (GUID) of the user to be subscribed.
Response name:	SubscribeUserPresenceResponse	
Response elements:	The operation does not return any results. The SOAP client can assume success if PSI does not return a SOAP fault message.	

5.4.2 SubscribeUserPresence Message

The `SubscribeUserPresence` message carries a `SubscribeUserPresence` operation request and its parameters from a SOAP client to PSI.

The following example presents the `SubscribeUser` message, which adds the user to the subscribed users list of the session. The subsequent `GetPresenceChanges` call returns the initial (current) presence state for the user.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<SubscribeUserPresence xmlns="http://sap.com/bcm/PSI">
 <sessionId>CF24E8FC82C34FED985BADE706CD209E</sessionId>
 <userId>648E588B1B0C439289627AF79869CFC2</userId>
 <evtFlags>All</evtFlags>
</SubscribeUserPresence>
</soap:Body>
</soap:Envelope>
```

5.4.3 SubscribeUserPresenceResponse Message

The `SubscribeUserPresenceResponse` message carries results of the `SubscribeUserPresence` operation from PSI back to a SOAP client. Note that a successful operation does not return any data elements.

Message example:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<SubscribeUserPresenceResponse xmlns="http://sap.com/bcm/PSI" />
</soap:Body>
</soap:Envelope>
```

5.5 UnsubscribeUserPresence Operation

5.5.1 Definition

Purpose:	The UnsubscribeUserPresence operation removes the presence subscription (ends presence monitoring) for the given user.	
Request name:	UnsubscribeUserPresence	
Request elements:	SessionID	Specifies the session ID retrieved from the CreateSession call.
	UserId	Sinch Contact Pro ID (GUID) of the user to be unsubscribed.
Response name:	UnsubscribeUserPresenceResponse	
Response elements:	The operation does not return any results. The SOAP client can assume success if PSI does not return a SOAP fault message.	

5.5.2 UnsubscribeUserPresence Message

The `UnsubscribeUserPresence` message carries an `UnsubscribeUserPresence` operation request and its parameters from a SOAP client to PSI.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<UnsubscribeUserPresence xmlns="http://sap.com/bcm/PSI">
<sessionId>3FFC7072DE8E4E5BA5567D8B8337325F</sessionId>
<userId>854BD9AB94934813AD6200F239480BDB</userId>
</UnsubscribeUserPresence>
</soap:Body>
</soap:Envelope>
```

5.5.3 UnsubscribeUserPresenceResponse Message

The `UnsubscribeUserPresenceResponse` message carries results of an `UnsubscribeUserPresence` operation from a PSI back to a SOAP client.

Message example:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<UnsubscribeUserPresenceResponse xmlns="http://sap.com/bcm/PSI" />
</soap:Body>
</soap:Envelope>
```

5.6 GetPresenceChanges Operation

5.6.1 Definition

Purpose:	The GetPresenceChanges operation is used to retrieve notifications of presence changes for all subscribed users in the session.	
Request name:	GetPresenceChanges	
Request elements:	SessionID	Specifies the session ID retrieved from the CreateSession call.
	Timeout	Specifies the time PSI waits for presence changes if there are

		currently no pending notifications to be returned. The value is specified in milliseconds. The default waiting time for presence changes is 0 (that is, immediate return). The requested timeout value must not be smaller than 0 ms and not greater than 60000 ms.
Response name:	GetPresenceChangesResponse	
Response elements:	The operation returns an array of PresenceChangeNotification, each element of the array specifies the changed presence state for a user. The contents of the PresenceChangeNotification element are described here.	

5.6.2 GetPresenceChanges Message

The `GetPresenceChanges` message carries a `GetPresenceChanges` operation request and its parameters from a SOAP client to PSI.

Message example:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<GetPresenceChanges xmlns="http://sap.com/bcm/PSI">
<sessionId>CF24E8FC82C34FED985BADE706CD209E</sessionId>
<timeOut>5000</timeOut>
</GetPresenceChanges>
</soap:Body>
</soap:Envelope>
```

5.6.3 GetPresenceChangesResponse Message

The `GetPresenceChangesResponse` message carries results of the `GetPresenceChanges` operation from PSI back to a SOAP client.

An example of the `GetPresenceChangesResponse` message:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body><GetPresenceChangesResponse xmlns="http://sap.com/bcm/PSI">
<GetPresenceChangesResult>
```

```
 <PresenceNotification>
  <UserId>648E588B1B0C439289627AF79869CFC2</UserId>
  <PresenceProfile>02770429F80BDF1191AD001CC4D98816</PresenceProfile>
  <EndTime>9999-12-31T23:59:59.9999999</EndTime>
  <LoginState>Logged_In</LoginState>
  <CallState>No_Change</CallState>
  <ServiceState>No_Change</ServiceState>
 </PresenceNotification>
</GetPresenceChangesResult>
</GetPresenceChangesResponse>
</soap:Body>
</soap:Envelope>
```

5.7 SetUserPresence Operation

5.7.1 Definition

Purpose:	The SetUserPresence operation is used to set the Sinch Contact Pro user presence for the given Sinch Contact Pro user.	
Request name:	SetUserPresence	
Request elements:	SessionID	Specifies the session ID retrieved from the CreateSession call.
	UserId <UserID>	Specifies the ID of the user whose presence is to be set.
	PresenceType <PresenceType>	Specifies the new presence for the user. Contents of the PresenceType structure are described here.
Response name:	SetUserPresenceResponse	
Response elements:	The operation does not return any results. The SOAP client can assume success if PSI does not return a SOAP fault message.	

5.7.2 SetUserPresence Message

The `SetUserPresence` message carries a `SetUserPresence` operation request and its parameters from a SOAP client to PSI.

Message example:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<soap:Body><SetUserPresence xmlns="http://sap.com/bcm/PSI">
<sessionId>B0D164C8A39D4F87B41B028BFA70EB54</sessionId>
<userId>648E588B1B0C439289627AF79869CFC2</userId>
```

```
<presence>
 <PresenceProfile>04770429F80BDF1191AD001CC4D98816</PresenceProfile>
 <EndTime>9999-12-31T23:59:59.9999999</EndTime>
 <ServiceState>No_Change</ServiceState>
 <CallState>No_Change</CallState>
</presence>
</SetUserPresence>
</soap:Body>
</soap:Envelope>
```

5.7.3 SetUserPresenceResponse Message

The `SetUserPresenceResponse` message carries results of the `SetUserPresence` operation from a PSI back to a SOAP client.

Message example:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<SetUserPresenceResponse xmlns="http://sap.com/bcm/PSI" />
</soap:Body>
</soap:Envelope>
```

5.8 SubscribeConfigChanges Operation

5.8.1 Definition

Purpose:	The SubscribeConfigChanges operation can be used to specify the presence synchronization groups whose configuration changes interest the PSI user. PSI filters the informed configuration changes to the users belonging to the given groups. If no subscription for the configuration changes exists, PSI reports changes to all users and groups in the GetConfigurationChanges operation.	
Request name:	SubscribeConfigChanges	
Request elements:	SessionID	Specifies the session ID retrieved from the CreateSession call.
	Array of Presence Synchronization Tokens <PSToken>	Specifies the target Sinch Contact Pro user groups the caller is interested in. Given PSTokens identify the groups in Sinch Contact Pro. One or several PSTokens can be provided. The maximum number of PSTokens per one call is currently 64.

Response name:	SubscribeConfigChangesResponse
Response elements:	The operation does not return any results. The SOAP client can assume success if PSI does not return a SOAP fault message.

5.8.2 SubscribeConfigChanges Message

The `SubscribeConfigChanges` message carries a `SubscribeConfigChanges` operation request and its parameters from a SOAP client to PSI.

Message example:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<SubscribeConfigChanges xmlns="http://sap.com/bcm/PSI">
<sessionId>AAD3C184E4D240BF8F12446F6C25215D</sessionId>
 <PSToken>
  <string>basic</string>
 </PSToken>
</SubscribeConfigChanges>
</soap:Body>
</soap:Envelope>
```

5.8.3 SubscribeConfigChangesResponse Message

The `SubscribeConfigChangesResponse` message carries results of a `SubscribeConfigChanges` operation from a PSI back to a SOAP client.

Message example:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<SubscribeConfigChangesResponse xmlns="http://sap.com/bcm/PSI" />
</soap:Body>
</soap:Envelope>
```

5.9 GetConfigurationChanges Operation

5.9.1 Definition

Purpose:	The GetConfigurationChanges operation is used to retrieve notifications of configuration changes for users or groups relevant for the session.	
Request name:	GetConfigurationChanges	
Request elements:	SessionID	Specifies the session ID retrieved from the CreateSession call.
Response name:	GetConfigurationChangesResponse	
Response elements:	The operation returns an array of ConfigurationChange elements, each element of the array speficies a configuration change for a user or a group. The contents of the ConfigurationChange element are described here.	

5.9.2 GetConfigurationChanges Message

The `GetConfigurationChanges` message carries a `GetConfigurationChanges` operation request and its parameters from a SOAP client to PSI.

Message example:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<GetConfigurationChanges xmlns="http://sap.com/bcm/PSI">
<sessionId>CF24E8FC82C34FED985BADE706CD209E</sessionId>
</GetConfigurationChanges>
</soap:Body>
</soap:Envelope>
```

5.9.3 GetConfigurationChangesResponse Message

The `GetConfigurationChangesResponse` message carries results of the `GetConfigurationChanges` operation from PSI back to a SOAP client.

Message example:

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<GetConfigurationChangesResponse xmlns="http://sap.com/bcm/PSI">
<GetConfigurationChangesResult>
 <ConfigurationChange>
  <ObjectId>basic</ObjectId>
  <ObjectType>PSTOKEN</ObjectType>
  <ChangeId>MODIFIED</ChangeId>
 </ConfigurationChange>
</GetConfigurationChangesResult>
</GetConfigurationChangesResponse>
</soap:Body>
</soap:Envelope>
```

5.10 EndSession Operation

5.10.1 Definition

Purpose:	The EndSession operation deletes the previously created session between PSI and its user.	
Request name:	EndSession	
Request elements:	SessionID	Specifies the session ID retrieved from the CreateSession call.
Response name:	EndSession	
Response elements:	The operation does not return any results. The SOAP client can assume success if PSI does not return a SOAP fault message.	

5.10.2 EndSession Message

The `EndSession` message carries an `EndSession` operation request and its parameters from a SOAP client to PSI.

Message example:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<EndSession xmlns="http://sap.com/bcm/PSI">
<sessionId>CF24E8FC82C34FED985BADE706CD209E</sessionId>
</EndSession>
</soap:Body>
</soap:Envelope>
```

5.10.3 EndSessionResponse Message

The `EndSessionResponse` message carries results of the `EndSession` operation from a PSI back to a SOAP client.

Message example:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<EndSessionResponse xmlns="http://sap.com/bcm/PSI" />
</soap:Body>
</soap:Envelope>
```

5.11 PSI Error Responses and Their Meanings

The following table contains possible errors returned by PSI.

Fault String	Meaning
Service failed to create a response	Fault code: Server An unexpected error occurred while the server tried to create the response. The error needs to be investigated from various system logs.
User not authorized to perform requested operation	Fault code: Client The requesting user has not the sufficient rights to perform the requested action.
Invalid GUID format of some provided user GUID	Fault code: Client Invalid format of some provided input GUID
Maximum simultaneous session count reached - cannot create more sessions	Fault code: Client The maximum number of simultaneous sessions is reached.
Invalid session ID or session expired	Fault code: Client The given session ID does not specify a session or the session has expired due to inactivity timeout
No tokens to process	Fault code: Client No PSTokens given in the GetUsers or SubscribeConfigChanges operations
Maximum number of tokens is 64	Fault code: Client Too many PSTokens given in the GetUsers or SubscribeConfigChanges operations
Invalid user id	Fault code: Client The given user ID does not exist or does not belong to the users in the session
Invalid DateTime provided	Fault code: Client The given DateTime is invalid (the SetUserPresence operation)
No service state or profile in input	Fault code: Client

	The given PresenceType is invalid in the SetUserPresence operation – either profile or service state needs to be specified
Invalid profile id	Fault code: Client The given presence profile ID is invalid in the SetUserPresence operation
Timeout value needs to be between 0 and 60000	Fault code: Client The given timeout value is invalid in the GetPresenceChanges operation

Example error:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Client</faultcode>
      <faultstring>Invalid session id or session expired</faultstring>
      <detail />
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

6 PSI Use Case Scenarios

This chapter provides some hints regarding how the methods of PSI could be used in practice.

6.1 Typical PSI Request Flow

The following operations normally take place during the PSI use:

1. CreateSession – this is essential in order to use the other operations
2. GetPresenceProfiles – caller retrieves the defined profiles
3. GetUsers – caller retrieves the list of users whose presence is to be monitored
4. SubscribeUserPresence – caller indicates its interest of the presence for certain users
5. GetPresenceChanges – caller executes this operation continuously in order to receive changes in presence for the subscribed users
6. SetUserPresence – caller sets the presence state for a user (for example a change in the external system is to be mirrored in the Sinch Contact Pro side)
7. GetConfigurationChanges – caller periodically checks possible changes in the Sinch Contact Pro side in order to refresh its configuration (for example to start subscribing new users or to update the profile list)
8. EndSession – caller ends the presence monitoring

6.2 Sample Code Using PSI

The following examples show the PSI use from an imaginary and non-functional C# application. The samples use PSI via a Web Reference object which is created automatically by the Add reference operation in Microsoft Visual Studio.

Initialization and session creation:

```
// PSI web reference
PSI.PSI P;

// session ID
```

```
String sessionId;

// Web service url
string PSIUrl = http://172.16.0.1/PSI/Service.asmx;

P = new PSI.PSI();

// Set user and password for authentication
System.Net.CredentialCache crc = new System.Net.CredentialCache();
System.Net.NetworkCredential cred = new System.Net.NetworkCredential(“PSIUser”, “pa55X9826”)
crc.Add(new System.Uri(PSIUrl), "Basic", cred);
P.Credentials = crc;

// Create the PSI session and store session id
PSI.CreateSessionResult csr;
csr = P.CreateSession();
sessionId = csr.SessionID;

// Fetch presence profiles
PSI.PresenceProfileType[] ap = P.GetPresenceProfiles();
foreach (PSI.PresenceProfileType pt in ap)
{
  Console.WriteLine("Profile id=" + pt.ProfileID + " name=" + pt.ProfileName);
}
```

Fetch users to be monitored and subscribe their presence:

```
string[] atokens = new string[1];
atokens[0] = "basic";

PSI.UserType[] au = P.GetUsers(sess,atokens);
foreach (PSI.UserType ut in au)
{
  Console.WriteLine("Subscribing presence for:");
  Console.WriteLine("User " + ut.BcmID + " / " + ut.ExtID);
  Console.WriteLine("--" + ut.FirstName + " " + ut.SurName);
  Console.WriteLine("-- PSToken group:" + ut.PSToken);

  P.SubscribeUserPresence(sess, ut.BcmID, PSI.EventFlags.All);

}
```

Continuously (for example in a separate thread) get the presence changes for the subscribed users:

```
void PresenceThread()
{
  while (true)
  {
    PSI.PresenceChangeNotification[] apn = P.GetPresenceChanges(sessionId, 5000);
    foreach (PSI.PresenceChangeNotification pcn in apn)
    {
      Console.WriteLine("State for " + pcn.UserId);
      Console.WriteLine("Profile:" + pcn.PresenceProfile);
      Console.WriteLine("Login:" + pcn.LoginState);
      Console.WriteLine("Service:" + pcn.ServiceState);
      Console.WriteLine("Call:" + pcn.CallState);
    }
  }
}
```

Check configuration changes every now and then:

```
void CheckConfigChanges()
{
  PSI.ConfigurationChange[] acc = P.GetConfigurationChanges(sess);
  foreach (PSI.ConfigurationChange cc in acc)
  {
    Console.WriteLine("Config change:");
    Console.WriteLine("Change Id: " + cc.ChangeId);
    Console.WriteLine("Object type:" + cc.ObjectType);
    Console.WriteLine("Object Id:  " + cc.ObjectId);
  }
}
```

Set presence for a user to synchronize the presence in an external system:

```
void SetProfile(string userId,string profileId)
{
  PSI.PresenceType pt = new PSI.PresenceType();
  pt.CallState = PSI.UserCallState.No_Change;
  pt.ServiceState = PSI.UserServiceState.No_Change;
  pt.EndTime = DateTime.MaxValue;
  pt.PresenceProfile = profileID;
  P.SetUserPresence(sessionId, userId, pt);
}
```