

eHF-UserManagement

Web Service Documentation

WebServiceDocumentation

WebServiceDocumentationVersion 1.0 | Status Approved
Security Level Public

Imprint

InterComponentWare AG
Industriestraße 41
69190 Walldorf, Germany
Tel.: +49 (0) 6227 385 0
Fax: +49 (0) 6227 385 199

Document Version: 1.0
Document Language: en (US)
Security Level: Public

Last change on: 9/22/2008
Editorial Staff: BAS

Table of Contents

1	User Management Web Service.....	5
1.1	Terminology	5
1.1.1	User	5
1.2	Overview	5
1.2.1	Web service Description:.....	5
1.2.2	impl:UsermgntXto	7
1.2.3	Impl:User	7
1.2.4	Impl:Role.....	7
1.2.5	Impl:UserIdentifier	8
1.2.6	Impl:Person	8
1.2.7	Impl:Address.....	10
1.2.8	Impl:BankAccount.....	11
1.2.9	Impl:Telecom	12
1.2.10	Impl:Payment.....	12
1.2.11	Impl:CreditCard	12
1.2.12	Impl:PageQualifier	13
1.2.13	Impl:PagedPersonResult.....	13
1.3	Method description	13
1.3.1	createUser	13
1.3.2	createUserWithPassword.....	14
1.3.3	updateUser	15
1.3.4	deleteUser	16
1.3.5	createPerson	17
1.3.6	createPersonWithPassword	18
1.3.7	updatePerson	19
1.3.8	updatePersonWithPassword	20
1.3.9	deletePerson	21
1.3.10	findUserByUserIdentifier.....	21
1.3.11	findUserByGuid	22

1.3.12	findPersonByUserIdentifiers	22
1.3.13	findPersonByUserGuid	23
1.3.14	findPersonByGuid.....	24
1.3.15	findPersonsByCriteria	24
1.3.16	findPersonsByCriteriaPaged	26
1.3.17	findPersonsByExplicitCriteria	26
1.3.18	findPersonsByExplicitCriteriaPaged.....	28
1.3.19	findLoginPolicyByName.....	29
1.3.20	findLoginPolicies.....	29
1.3.21	findSyntaxPolicyByName	29
1.3.22	findSyntaxPolicies	30
1.3.23	findRolesByNameRecurse	30
1.3.24	changePassword	31
1.3.25	resetPassword	32
1.3.26	findCurrentUser	32
1.3.27	findCurrentPerson	33
1.3.28	findPersonsByRolesIncludeAddresses	33
1.3.29	addIdentifier	34
1.3.30	removeIdentifier	35
1.3.31	resetSecret	35
1.3.32	verifySecret.....	36

1 User Management Web Service

1.1 Terminology

1.1.1 User

A User object represents a user of the application. An instance of this class is the central starting point of references to all user related data like roles, user identifiers and the user's password. A user is an entity with the ability to authenticate. Users, that are marked inactive, are not allowed to log into the application.

User Identifier

The primary use case for user identifier objects is to uniquely identify a user. A user can have several user identifiers for different contexts (authentication, data import).

Person

A person object holds all the personal data. A person object is without the possibility to authenticate. Typically a person is associated with a user of the application. In this case an instance of the Person class references one User object.

Role

Each user can act in multiple roles. Roles are arranged hierarchically. Therefore, by assigning a role to a user, the user implicitly gets all the role's parent roles assigned.

1.2 Overview

1.2.1 Web service Description:

The User Management web service allows you to create and administrate persons and users interacting with the application.

Usage Notes:

Administration of person data

The central point of the user management is the person type. This web service provides CRUD services to maintain person data. A person itself provides information

of a customer like name, birth date and gender. Furthermore, additional attributes can be supplied to the person information as well:

multiple addresses

multiple communication contacts

multiple professions

payment information

bank account information

credit card information

user information

The methods `createPerson` and `createPersonWithPassword` can be used to create new persons. All the aforementioned information provided for the person object are then stored in the database. To participate in the application, a person must have related user information. It is recommended to provide user information with the person while creating it. Both methods only differ in the password attribute. In `createPersonWithPassword` the plain text password of the person's user is provided. When using `createPerson`, a new person (probably inclusive it's related user) is created without a password. In order to create an initial one-time-password the `resetPassword` method has to be called explicitly to generate, encode and assign a one-time-password to the person's user. A user's password can always be updated using `changePassword`. A person can be updated with `updatePerson` and `updatePersonWithPassword`. The latter does update the password as well as all other person information. The `deletePerson` method can be used to delete an existing person. To retrieve existing persons of the system, this web service provides multiple finder methods, all starting with `findPersonBy...`

Administration of user data

Each person must be related to a user to be able to interact with the system. It is recommended to use the person methods to create a person plus its user. System users are reasonably not related to a person and therefore, the person methods are not sufficient. For this case, the user management web service provides the methods `createUser` and `createUserWithPassword` to create a user without a correlating person plus its user identifiers and roles. The only difference again is the provision of the user's password. Updates can be performed similar to synchronously to the person update using `updateUser`. To delete an existing user, `deleteUser` must be executed. To retrieve existing users the web service offers finder methods starting with `findUserBy...`

Main method parameters:

Parameter	Description	Used By
Impl:Person	Holds all the personal data	findPersonsByCriteria, findPersonsByExplicitCriteria, createPerson, createPersonWithPassword, updatePerson, updatePersonWithPassword, deletePerson
Impl:User	The user of the application	createUser, updateUser, createUserWithPassword
Impl:Role	The role of a user or group	findPersonsByExplicitCriteria, findPersonsByRolesIncludeAddresses

All of the main parameters mentioned above extend the **UsermgntXto**.

1.2.2 impl:UsermgntXto

Attribute	Description	Type	Range
guid	The unique Globally Unique Identifier of this instance.	xsd:string	unique
domain	The domain of this instance. Objects within the same domain are grouped together logically.	xsd:string	

1.2.3 Impl:User

Attribute	Description	Type	Range
active	Specifies whether this user is active or not	xsd:Boolean	
system	Specifies if this user is a system user. System users are not related to a person instance.	xsd:Boolean	
roles	The roles of a user.	ArrayOfRole	
identifiers	A set of all user identifiers	ArrayOfUserIdentifier	

1.2.4 Impl:Role

Attribute	Description	Type	Range
-----------	-------------	------	-------

name		xsd:string	ICW-PHR- USER-ROLE, mandatory, unique
parent	The parent role	impl:role	

1.2.5 Impl:UserIdentifier

A user identifier object can be associated with a *SyntaxPolicy* object. In this case the *value* property of the user identifier object must be in compliance with this syntax policy. An instance of this class is always associated with only one *User* object.

Constraints:

pairs of user identifier type, user identifier value must be unique. User identifiers having the same type and value are not allowed.

Attribute	Description	Type	Range
type	Defines the type of this user identifier. Any user can have multiple user identifiers of different types	xsd:string	ICW-PHR- USER- IDENTIFIER, mandatory, not modifiable
value	The value field holds the value of the user identifier.	xsd:string	Max length 255
active	Specifies whether this identifier is active or not	xsd:Boolean	

1.2.6 Impl:Person

Attribute	Description	Type	Range
gender		impl:gender	EXT-GEN- PERSON- GENDER, manda- tory

birthName, lastName, firstName, middleName, secondName		xsd:string	Max length 255
namePrefix, nameSuffix	The optional name prefix/suffix which may be put before/after the first/last name	xsd:string	Max length 255
title		xsd:string	Max length 255
birthdate		xsd:dateTime	
addresses	The addresses of this person. None of these addresses act as the primary address. One special address (e.g. the home address) can be declared as primary address in the primaryAddress field	ArrayOfAddress	Setting an address both as primary and part of the additional addresses is strictly prohibited.
bankAccount		impl:BankAccount	
creditCard		impl:CreditCard	
payment		impl:Payment	
primaryAddress		impl:Address	Setting an address both as primary and part of the additional addresses is strictly prohibited
primaryProfessions	The primary professions describe the main professions this person practices. Additional, non primary professions are maintained by the field <i>professions</i> .	ArrayOfProfession	Setting professions both as primary and part of the additional professions is strictly prohibited
primaryTelecom	One special contact (e.g. the mobile phone contact)	impl:Telecom	
professions		ArrayOfProfession	

telecoms	Defines the telecommunications contact data of this person. None of these telecommunication contacts acts as the primary telecommunication contact.	ArrayOfTelecom	Setting a telecommunication contact both as primary and part of the additional contacts is strictly prohibited
user	Defines the user which is associated with this person. If the user is associated to a person this person is able to interact with the system by means of the user. The user itself holds additional information, for example its roles and user identifiers.	impl:User	

1.2.7 Impl:Address

Attribute	Description	Type	Range
useCode	Describes the use of this address. It must be differentiated between private or business (e.g. clinic, practice or pharmacy address) addresses. If it is a home address, the use code must be set to "H". If the address is a workplace address, "WP" must be provided.	Xsd:string	EXT-GEN-ADDRESS-USE
street		xsd:string	Max length 255
city		xsd:string	Max length 255
zipCode	It covers all country-specific zip codes. If a country works with additional zip code information, these can be defined in the <i>zipCodeExtension</i> field.	xsd:string	Max length 255
zipCodeExtension	Several countries have a zip code system which consists of a zip code plus additional information. This additional information can be placed here.	xsd:string	Max length 255
country		xsd:string	EXT-GEN-ADDRESS-COUNTRY

region	Defines the region of the address. It could for example be a state in the USA or a canton in Switzerland.	xsd:string	C-ADDRESS-REGION
line1	Allows additional information of the address which does not fit in one of the other address fields. For example: "entrance is on the backside of the building". If the length limitation is probably violated, the <i>line2</i> can be used for further description	xsd:string	Max length 255
line2	Can be used for additional information in case the information exceeds the limitation of <i>line1</i> . At first, please use <i>line1</i> for additional information and <i>line2</i> only if the character space does not suffice.	xsd:string	Max length 255
corpus		xsd:string	Max length 255
flat	The flat allows a more precise description of the address. It specified the flat related to the postal address. One example can be "first floor, entrance on the right side".	xsd:string	Max length 255

1.2.8 Impl:BankAccount

Attribute	Description	Type	Range
holderName, instituteName, instituteNumber, number		xsd:string	Max length 255
iban	The International Bank Account Number of this account.	xsd:string	Max length 255
bic	The Bank Identifier Code (BIC) of the bank holding this bank account.	xsd:string	Max length 255

1.2.9 Impl:Telecom

Attribute	Description	Type	Range
code	Defines the type of telecommunication contact. Valid telecommunication contact codes are for example "TEL" for telephone, "TELMO" for mobile phone or "EMAIL" for email.	xsd:string	ICW-GEN-TELECOM-CODE-DE
useCode	Defines whether the telecommunication contact is a private or business contact. A private telecommunication is defined by the value "H" for home contact. If it is a business contact, "WP" must be the value (stands for workplace).	xsd:string	EXT-GEN-TELECOM-USE

1.2.10 Impl:Payment

Attribute	Description	Type	Range
mode	The payment mode defines the method of payment chosen by the related person. Examples of payment modes are "bank transfer" and "credit card".	xsd:string	ICW-GEN-PAYMENT-MODE

1.2.11 Impl:CreditCard

Attribute	Description	Type	Range
type	Describes the credit card type respectively the issuing company (for example "VISA" and "MasterCard")	xsd:string	EXT-GEN-PAYMENT-CARD-TYPE
number		xsd:string	Max length 255
holder		xsd:string	Max length 255
validity	Describes the validity of the credit card. The format must fit the common validity schema for credit cards (e.g. "07/12" for July 2012).	xsd:string	Max length 28

The following parameter does **not** extend UsermngtXto

1.2.12 Impl:PageQualifier

Attribute	Description	Type	Range
id	The transient id of this instance used to retrieve consecutive pages of the same result from the internal cache	xsd:string	
pageNumber	The page number to retrieve	xsd:integer	
pageSize	The size of the pages to retrieve	xsd:integer	
locale	The locale to use	xsd:string	

The following is a return parameter

1.2.13 Impl:PagedPersonResult

Attribute	Description	Type	Range
pageQualifier	PageQualifier specifying the shape of the result	Impl:PageQualifier	
totalNumberOfObjects	Total number of persons found	xsd:integer	
totalNumberOfPages	Total number of pages to retrieve	xsd:integer	
objects	The return values of the page as specified in pageQualifier	arrayOfPersons	

1.3 Method description

1.3.1 createUser

Description:

Creates a new user. The user's identifiers are used for authentication and should be provided. To be authorized to perform operations, the user must be assigned to at least one role. After successful creation of the user, its GUID is returned.

Usage Notes:

- A hereby created user is never related to a person. If a relation is intended, use `createPerson` instead and provide this user to the person instance.
- Users without user identifiers and roles are not able to participate in the application because of missing privileges. To update user information use `updateUser`.
- The created user does not have a password. To provide one, the method `changePassword` can be used.
- If the user needs to have a user identifier for client certificate authentication (type `CERTIFICATE_OWNER_ID`), the certificate information must be provided as user identifier value and must be base64-encoded.

Arguments:

Parameter	Description	Data Type	Usage
User	The data for the user to be created	Impl:User	R

Returns:

Description	Data Type
GUID of the newly created user	xsd:string

1.3.2 createUserWithPassword

Description:

Creates a new user with given password. The user's identifiers are used for authentication and should be provided. To be authorized to perform operations, the user must be assigned to at least one role. The password must be provided in plain text and gets encoded and assigned to the user before persisting it. The user type does not provide a password field because changing passwords requires additional security checks. After successful creation of the user, its GUID is returned.

Usage Notes:

- A hereby created user is never related to a person. If this is not intended, use `createPersonWithPassword` instead and provide this user to the person instance.

- Users without user identifiers and roles are not able to participate in the application because of missing privileges. To update user information use `updateUser`.
- If the user needs to have a user identifier for client certificate authentication (type `CERTIFICATE_OWNER_ID`), the certificate information must be provided as user identifier value and must be base64-encoded.

Arguments:

Parameter	Description	Data Type	Usage
User	The data for the user to be created	Impl:User	R
password	The password of the user	xsd:string	R

Returns:

Description	Data Type
the GUID of the newly created user	xsd:string

Error codes/Exceptions:

Error code/Exception	Reason
SyntaxPolicyException	is thrown if the password doesn't match the password syntax policy

1.3.3 updateUser

Description:

Modifies data of the given user. The user with the same GUID as the given user is updated. The update is performed on the whole object graph of the user, so for example roles and user identifiers can be modified as well. The only exception here is the manipulation of the user's password. Here, `changePassword` must be used because changing passwords requires additional security checks.

Usage Notes:

-The `strict` value determines how `null` values are handled. If it is set to `true`, all values of the object graph, including `null`, overwrite the corresponding values of the persistent model. By setting it to `false`, `null` values in the input graph are ignored.

- The `strict` flag must be handled with care. `false` must be used if you want to update only a few values, for example a single user identifier, without having to specify the complete graph (roles etc.). Be aware that when using `strict` enabled, you have to provide all values you want the user to have or they will be overwritten with a `null` value.

- To update the user's password, you have to call `changePassword` instead of this method.

Arguments:

Parameter	Description	Data Type	Usage
User	the new user data	Impl:User	R
<code>strict</code>	if <code>true</code> all values in the object graph are updated (including <code>null</code> values). If <code>false</code> , only the values which are not <code>null</code> get updated	xsd:boolean	R

Returns: none

1.3.4 deleteUser

Description:

Deletes the user which matches the given user GUID. By deleting a user, all its user identifiers and its password are deleted as well. The roles are not deleted since they may be referenced by other users.

Usage Notes:

Probably, the user is related to a person. In that case, it is recommended to not delete the user and delete the person instead (which implicitly deletes the related user). To do so, please use `deletePerson` instead. A person without a related user is not able to participate in the application any more.

Arguments:

Parameter	Description	Data Type	Usage
User	the user to be deleted	Impl:User	R

Returns: none

1.3.5 createPerson

Description:

Creates a new person from the given object graph. This object graph may include bank account, credit card and payment information as well as addresses, telecommunication contacts and profession information. A person should always be related to a user. So it is necessary to provide the user information together with the person information at creation time. After successful creation of the person, its GUID is returned.

Usage Notes:

- Ensure, that the person to be created is provided with user information. Otherwise, the method call will fail because creation of a person without an associated user is prohibited.
- The provided user for this person is created without assigning any password. To create a person with user and password, you should use `createPersonWithPassword`.

Arguments:

Parameter	Description	Data Type	Usage
person	the person to be created	Impl:Person	R

Returns:

Description	Data Type
the GUID of the newly created person	xsd:string

1.3.6 createPersonWithPassword

Description:

Creates a new person with password from the given object graph. This object graph may include bank account, credit card and payment information as well as addresses, telecommunication contacts and profession information. A person should always be related to a user. So it is necessary to provide the user information together with the person information at creation time. The password must be provided in plain text and gets encoded and assigned to the person's user before persisting it. The user type does not provide a password field because changing passwords requires additional security checks. After successful creation of the person, its GUID is returned.

Usage Notes:

- Ensure, that the person to be created is provided with user information. Otherwise, the method call will fail because creation of a person without an associated user is prohibited.

Arguments:

Parameter	Description	Data Type	Usage
person	the person to be created	Impl:Person	R
password	the password of the person's user	xsd:string	R

Returns:

Description	Data Type
the GUID of the newly created person	xsd:string

Error Codes/Exceptions:

Error code/Exception	Reason
SyntaxPolicyException	is thrown if the password doesn't match the password syntax policy

1.3.7 updatePerson

Description:

Modifies data of the given person. The person with the same GUID as the given person is updated. The update is performed on the whole object graph of the person, so for example addresses, professions or the user with its user identifiers can be modified as well. The only exception here is the manipulation of the user's password. Here, `changePassword` respectively `updatePersonWithPassword` must be used because changing passwords requires additional security checks.

The `strict` value determines how `null` values are handled. If it is set to `true`, all values of the object graph, including `null`, overwrite the corresponding values of the persistent model. By setting it to `false`, `null` values in the input graph are ignored. Direct person attributes (birth date, birth name, first name, second name, middle name, name prefix, name suffix, last name, gender and title) are not sensitive to the `strict` flag. These attributes get always overwritten, so it must be ensured that the values of these properties are provided as well.

Usage Notes:

Ensure, that the person to be updated is provided with user information. Otherwise, the method call will fail because the existence of a person without an associated user is prohibited.

- The `strict` flag must be handled with care. `false` must be used if you want to update only a few values, for example a single address, without having to specify the complete graph (professions etc.). Be aware that when using `strict` enabled, you have to provide all values you want the person to have or they will be overwritten with a `null` value.

Arguments:

Parameter	Description	Data Type	Usage
person	the person to be created	Impl:Person	R
strict	<code>strict</code> - if <code>true</code> all values in the object graph are updated (including <code>null</code> values). If <code>false</code> , only the values which are not <code>null</code> get updated	xsd:boolean	R

Returns: none

1.3.8 updatePersonWithPassword

Description:

Modifies data of the given person plus the person's user password. The person with the same GUID as the given person is updated. The update is performed on the whole object graph of the person, so for example addresses, professions or the user with its user identifiers can be modified as well.

The `strict` value determines how `null` values are handled. If it is set to `true`, all values of the object graph, including `null`, overwrite the corresponding values of the persistent model. By setting it to `false`, `null` values in the input graph are ignored. Direct person attributes (birth date, birth name, first name, second name, middle name, name prefix, name suffix, last name, gender and title) are not sensitive to the `strict` flag. These attributes get always overwritten, so it must be ensured that the values of these properties are provided as well.

Usage notes:

- Ensure, that the person to be updated is provided with user information. Otherwise, the method call will fail because the existence of a person without an associated user is prohibited.
- The `strict` flag must be handled with care. `false` must be used if you want to update only a few values, for example a single address, without having to specify the complete graph (professions etc.). Be aware that when using `strict` enabled, you have to provide all values you want the person to have or they will be overwritten with a `null` value.

Arguments:

Parameter	Description	Data Type	Usage
person	the person to be created	Impl:Person	R
password	the password of the person's user	xsd:string	R
strict	if <code>true</code> all values in the object graph are updated (including <code>null</code> values). If <code>false</code> , only the values which are not <code>null</code> get updated	xsd:boolean	R

Returns: none

Error Codes/Exceptions:

Error code/Exception	Reason
SyntaxPolicyException	is thrown if the password doesn't match the password syntax policy

1.3.9 deletePerson

Description:

Deletes the person which matches the given person GUID. By deleting a person, all its addresses, telecommunication contacts, professions and the corresponding user instance are deleted as well.

Usage Notes:

For some users (e.g. system users) there exists no corresponding person. In that case, `deleteUser` must be invoked to delete this user instance.

Arguments:

Parameter	Description	Data Type	Usage
person	the person to be deleted	Impl:Person	R

Returns: none

1.3.10 findUserByUserIdentifier

Description:

Finds the user object that is associated with the given identifier. If no user for the given user identifier exists, `null` is returned

Arguments:

Parameter	Description	Data Type	Usage
identifier	the user identifier that identifies the user that is associated with the person to be found	Impl:userIdentifier	R

Returns:

Description	Data Type
The user instance found including all objects referenced by this Person object (the user instance, addresses, telecoms, etc.)	impl:user

1.3.11 findUserByGuid

Description:

Finds the user instance which is identified by the given GUID. If no user for the given GUID exists, `null` is returned.

Arguments:

Parameter	Description	Data Type	Usage
guid	the GUID that identifies the user to be found	xsd:string	R

Returns:

Description	Data Type
The user object found identified by the given GUID including all objects that are referenced by this user object (that is roles, user identifiers, etc.)	impl:user

1.3.12 findPersonByUserIdentifiers

Description:

Finds the person instance that is associated with the user that is identified by the given user identifiers. If no user for the given user identifier exists, `null` is returned.

Arguments:

Parameter	Description	Data Type	Usage
identifiers	the user identifiers that identifies the user that is associated with the person to be found	Impl:userIdentifier	R

Returns:

Description	Data Type
The person instance found including all objects referenced by this Person object (the user instance, addresses, telecoms, etc.)	impl:person

1.3.13 findPersonByUserGuid

Description:

Finds the person instance identified by the given `userGuid`. If no user for the given user GUID exists, `null` is returned.

Usage notes:

- This finder only returns the found person instance containing person, user and address data. All other information (for example professions and telecommunication contacts) is not provided. If these are desired, you have to choose another alternative finder implementation, e.g. `findPersonByGuid`.

Arguments:

Parameter	Description	Data Type	Usage
userGuid	the GUID of the user that is associated with the person to be found	xsd:string	R

Returns:

Description	Data Type
-------------	-----------

a person instance linked to a user and address instance or <code>null</code> if no user with given <code>userGuid</code> exists.	impl:person
--	-------------

1.3.14 findPersonByGuid

Description:

Finds the person with the given GUID. If no person for the given person GUID exists, `null` is returned.

Arguments:

Parameter	Description	Data Type	Usage
guid	the guid of the person to be found	xsd:string	R

Returns:

Description	Data Type
the person or <code>null</code> if no person is found for the given guid	impl:Person

1.3.15 findPersonsByCriteria

Description:

Finds all person instances matching the provided person criteria. Criteria can be specified by the `person` argument. By assigning more information to this person the results will be narrowed. This finder supports the following person attributes as criteria:

first name, second name, middle name, last name, name prefix, name suffix, birth name, birth date, title and gender of the person

city, zip code, zip code extension, use code, street, country, region, corpus, flat, line1 and line2 of the person's address

name of the person's profession

type and value of the user identifier of the person's corresponding user

role names of the person's corresponding user

The attribute values can be fully qualified expression or expressions using *-wildcards. For example, if the last name of the criteria person is set to "K*", all persons are returned whose last name starts with "K". This allows mature criteria definitions which are especially useful for person searches.

All string criteria values of Person, Address and Profession are case insensitive, e.g. "maier" matches "Maier" as well as "MAIER" etc.

Usage Notes:

A person's user may have multiple professions and addresses. But the criteria person only allows the definition of one profession respectively address criterion. If more than one is provided, only the first one will be taken

Unlike professions and addresses, multiple roles can be defined and all get evaluated by this method. The only thing to care about is that only persons are returned whose user is directly assigned to the criterion role. For example, a search for all persons in role `USR` will not return the persons which are in roles `NPR` or `PRF` because they are child nodes of the `USR` role according to the role hierarchy.

Adding any object to the criteria that is not mentioned above (like for example user information, telecommunication contacts, etc.) is not supported by this method and thus not evaluated by this finder method.- If no criteria is defined, this method will return all person instances. So this method should only be used when criteria are provided to narrow the search. Otherwise, the method will return extremely huge result sets.

This finder does not return the complete object graph of each person. The returned persons only contain person data like name and birth. If the complete object graph is desired, `findPersonsByExplicitCriteria` should be used instead.

Arguments:

Parameter	Description	Data Type	Usage
person	the person instance holding the finder criteria	Impl:Person	R

Returns:

Description	Data Type
an array of <code>Person</code> instances matching the given criteria	<code>ArrayOfPerson</code>

1.3.16 `findPersonsByCriteriaPaged`

Description:

Paging variant of `findPersonsByCriteria` (see 1.3.15)

Usage Notes:

Paging variant of `findPersonsByCriteria` (see 1.3.15)

In order to retrieve consecutive pages the `pageNumber` attribute of the returned `PageQualifier` has to be incremented before the next request.

Arguments:

Parameter	Description	Data Type	Usage
<code>person</code>	the person instance holding the finder criteria	<code>Impl:Person</code>	R
<code>pageQualifier</code>	A <code>PageQualifier</code> specifying the desired shape of the result	<code>Impl:PageQualifier</code>	R

Returns:

Description	Data Type
A <code>PagedPersonResult</code> containing the persons found as well as the <code>PageQualifier</code> used	<code>PagedPersonResult</code>

1.3.17 `findPersonsByExplicitCriteria`

Description:

Finds all person instances matching the provided person, address and role criteria. The `address` and `roles` arguments are optional and narrow the result set. This method returns all persons which match all the given criteria. Following attributes can be assigned as search criteria:

first name, second name, middle name, last name, name prefix, name suffix, birth name, birth date, title and gender of the person argument

city, zip code, zip code extension, use code, street, country, region, corpus, flat, line1 and line2 of the address argument

multiple role names of the roles argument

The attribute values can be fully qualified expression or expressions using wild-cards. For example, if the last name of the criteria person is set to "K*", all persons are returned whose last name starts with "K". This allows mature criteria definitions which are especially useful for person searches.

Usage Notes:

- Adding any object to the criteria that is not mentioned above (like for example user information, telecommunication contacts, etc.) is not supported by this method and thus not evaluated by this finder method.
- You get the most expanded result set if only an empty person instance is provided as first argument and the optional ones are `null`. Such a query should be avoided because than the method will return extremely huge result sets.
- In contrast to the `findPersonsByCriteria` this method returns the whole person object graph including all dependencies.
- For the person argument, only the person information (e.g. names) is evaluated when searching for persons. So setting of other dependent information (e.g. telecommunication contacts or user identifiers) is not recommended because it is not considered by this person search.

Arguments:

Parameter	Description	Data Type	Usage
person	a person object defining search criteria	Impl:Person	R
address	an address object defining search criteria (optional)	Impl:Address	O
roles	a role object defining search criteria	impl:role	O

Returns:

Description	Data Type
an array of persons matching the given criteria	ArrayOfPerson

1.3.18 findPersonsByExplicitCriteriaPaged

Description:

Paging variant of findPersonsByExplicitCriteria (see 1.3.17)

Usage Notes:

Paging variant of findPersonsByExplicitCriteria (see 1.3.17)

In order to retrieve consecutive pages the pageNumber attribute of the returned PageQualifier has to be incremented before the next request.

Arguments:

Parameter	Description	Data Type	Usage
person	the person instance holding the finder criteria	Impl:Person	R
address	an address object defining search criteria (optional)	Impl:Address	O
roles	a role object defining search criteria	impl:role	O
pageQualifier	A PageQualifier specifying the desired shape of the result	Impl:PageQualifier	R

Returns:

Description	Data Type
A PagedPersonResult containing the persons found as well as the PageQualifier used	PagedPersonResult

1.3.19 findLoginPolicyByName

Description:

Finds the login policy corresponding to the given unique name.

Arguments:

Parameter	Description	Data Type	Usage
name	the name of the desired login policy	xsd:string	R

Returns:

Description	Data Type
the login policy with the given name or null if no one exists with the given name	impl:LoginPolicy

1.3.20 findLoginPolicies

Description:

Finds all login policies defined within the system.

Returns:

Description	Data Type
an array with all found login policies	ArrayOfLoginpolicy

1.3.21 findSyntaxPolicyByName

Description:

Finds the syntax policy for the given name.

Arguments:

Parameter	Description	Data Type	Usage
name	the name of the syntax policy to be found	xsd:string	R

Returns:

Description	Data Type
the GUID of the newly created person	xsd:string

1.3.22 findSyntaxPolicies

Description:

Finds all syntax policies defined within the system.

Arguments: none

Returns:

Description	Data Type
an array with all found syntax policies	ArrayOfSyntaxPolicy

1.3.23 findRolesByNameRecurse

Description:

Finds all roles in a role hierarchy subtree. Since roles are arranged hierarchically, this method returns the role with the given role name and all its child roles. If the role with the given role name does not exist, an empty array is returned.

Arguments:

Parameter	Description	Data Type	Usage
-----------	-------------	-----------	-------

roleName	the name of the root node of role hierarchy subtree	xsd:string	R
----------	---	------------	---

Returns:

Description	Data Type
an array with all roles in the role hierarchy subtree	ArrayOfRole

1.3.24 changePassword

Description:

Modifies the password of the user with the given GUID. The method verifies that the given password value is in compliance with its associated syntax policy. If no syntax policy is associated with the user's password, the system's standard syntax policy is applied on the password value.

The new password must be provided in clear text and gets base64 encrypted.

If the password change is successful, all login status related values (`failedLoginCount`, `validLoginCount` and `expirationDate`) of the password are re-initialized.

Usage Notes:

The new password must match the password syntax policy.

Arguments:

Parameter	Description	Data Type	Usage
userGuid	the guid of the user whose password should be changed	xsd:string	R
password	the new password in clear-text	xsd:string	R

Returns: none

Error Codes/Exceptions:

Error code/Exception	Reason
SyntaxPolicyException	is thrown if the password doesn't match the password syntax policy
FinderException	if the user with the given GUID does not exist

1.3.25 resetPassword

Description:

Resets the password of the user identified by the given GUID. The password is generated by the application and is in compliance with the standard syntax policy and the one-time-password login policy configured for the application.

Arguments:

Parameter	Description	Data Type	Usage
userGuid	the guid identifying the user whose password is to be reseted	xsd:string	R

Returns:

Description	Data Type
the newly generated one-time password in clear text	xsd:string

1.3.26 findCurrentUser

Description:

Finds the user who is logged-in to this session. The full user instance with all its dependencies is returned.

Arguments: none

Returns:

Description	Data Type
-------------	-----------

the logged-in user instance	impl:user
-----------------------------	-----------

Error Codes/Exceptions:

Error code/Exception	Reason
SyntaxPolicyException	is thrown if the password doesn't match the password syntax policy

1.3.27 findCurrentPerson

Description:

Finds the person of the user who is logged-in to this session. The full person instance with all its dependencies is returned.

Usage Notes:

Because there exist users without a corresponding person, this method will return `null` if no such person exists to the current user.

Arguments: none

Returns:

Description	Data Type
the person including the assigned user and all assigned addresses	impl:person

1.3.28 findPersonsByRolesIncludeAddresses

Description:

Finds all person instances that are associated with a user which are assigned to at least one of the given roles.

Usage Notes:

The method returns the found persons object graphs including all address, user and role references, but no other references.

Arguments:

Parameter	Description	Data Type	Usage
roles	an array with the roles used to search for the persons	Impl:role	R

Returns:

Description	Data Type
the array with all found persons	ArrayOfPerson

1.3.29 addIdentifier

Description:

Adds the given user identifier to the user specified by the given guid.

Usage Notes:

The given user identifier must uniquely identify the user and therefore the identifier must be unique within the system.

Arguments:

Parameter	Description	Data Type	Usage
guid	the guid of the user the identifier should be added to	xsd:string	R
Identifier	the identifier to be added	Impl:userIdentifier	R

Returns: none

Error Codes/Exceptions:

Error code/Exception	Reason
AccessControlException	if the executing user has no permission to add this identifier
FinderException	if the user with the given GUID does not exist

1.3.30 **removelfdentifier**

Description:

Removes the given user identifier from the user specified by the given guid.

Usage Notes: -

Arguments:

Parameter	Description	Data Type	Usage
guid	the guid of the user the identifier should be removed from	xsd:string	R
Identifier	the identifier to be removed	Impl:userIdentifier	R

Returns: none

Error Codes/Exceptions:

Error code/Exception	Reason
AccessControlException	if the executing user has no permission to remove this identifier
FinderException	if the user with the given GUID does not exist

1.3.31 **resetSecret**

Description:

Resets the secret of the given type of the user identified by the given guid. The secret is generated by the application and is in compliance with the SyntaxPolicy and the LoginPolicy configured for the given secret type. The given digest is used to hash the secret.

Usage Notes: See the UserSecret class for supported digest algorithms. If the user does not have a secret of the given type, a new secret of that type is created and added to the user's secrets. See the SecretGenerator class for allowed secretCharacterSets.

Arguments:

Parameter	Description	Data Type	Usage
guid	the guid identifying the user whose secret is to be reset	xsd:string	R
secretType	the type of the secret to be reset	xsd:string	R
secretDigest	the digest used for the secret	xsd:string	R
secretCharacterSet	the character set for the secret	xsd:int	R

Returns: the generated secret

Error Codes/Exceptions:

1.3.32 **verifySecret**

Description:

Verifies the correctness of the given characters at the given positions against a user's secret.

Usage Notes:

Arguments:

Parameter	Description	Data Type	Usage
guid	the guid identifying the user whose secret is to be verified	xsd:string	R
secretType	the type of the secret to be verified	xsd:string	R

positions	the positions in the secret string that are to be verified	xsd:int	R
characters	a string consisting of the expected characters at the given positions in the secret. Example: If you expect the character 'b' on the first position of the secret and the character 'c' on the fourth position, then provide the string "bc" here.	xsd:string	R

Returns: true, if the given expected characters match with the secret at the given positions, false otherwise