Aqts_api Technical Reference

Introduction

The AQT Services API (aqts_api) is a .NET webservice that interfaces the web framework (coded in Python) with the AQT Services database. All code examples are in Python, using the suds SOAP client module.

All code examples are preceded by the following:

```
## Requires suds
from suds.client import Client

## URL that points to SOAP webservice
url = 'http://dev50/aqts_api/n_aqts_api.asmx?WSDL'

## Create suds client
client = Client(url)
```

Methods

addticketresponse

Add a response to an existing ticket.

Definition and Parameters

addticketresponse(xs:short is_id, xs:short ct_id, xs:string response_text)

is_id: The ticket number.

Ct_id: The contact ID of the person associated with the response.

Response_text: The actual text of the response.

Return

Returns a complex python object.

S_addticketresponse.retc: Return code.

0: Success.

-1: Database connectivity error.

-2: Error during insert.

Example

result = client.service.addticketresponse (12345, 1680, “Hello World.”)

print result.retc

authenticateuser

Authenticate the user based on username (email) and password.

Definition and Parameters

authenticateuser(xs:string email, xs:string password)

email: User's e-mail.

Password: User's password.

Return

Returns a complex python object.

S_authenticateuser.retc: Return code.

0: User authenticated.

-1: Retrieval error.

-2: Email not found.

-3: Password does not match.

S_authenticateuser.is_employee: Is User an AQT employee? True/false.

S_authenticateuser.aqts_id: AQT Services contact ID. Unique identifier.

S_authenticateuser.company_name: Name of the company associated with user.

S_authenticateuser.first_name: First name of user.

S_authenticateuser.last_name: Last name of user.

S_authenticateuser.position: Position title of user.

S_authenticateuser.password: User password.
S_authenticateuser.email: Email of user.

Example

```
result = client.service.authenticateuser("faym@aqtsolutions.com", "changeme")

print result.retc

print result.position

print result.company_name
```

createticket

Create a ticket in the AQT Services database.

Definition and Parameters

createticket(xs:int aqts_id, xs:string product, xs:string release, xs:string priority, xs:string issue_type, xs:string window, xs:string keywords, xs:string issue)

aqts_id: AQTS contact id.

product: Name of product, ie. "ATMS Core".

Release: Version, ie. "5.10"

Priority: "Moderate", "Severe", etc.

Issue_type: Type of issue. ie. "Problem", "Question".

Window: Window/Object where the issue is presenting.
Keywords: Keywords for the issue.

Issue: Issue description.

Return

Returns a complex python object.

S_createticket.retc: Return code.

0: Success.

-1: Database connectivity error.

-2: Error creating ticket.

S_createticket.is_id: Issue # of new ticket.

Example

result = client.service.createticket(11049, "ATMS Core", "5.00", "Serious", "Problem", "Course Maintenance", "keyword", "description")

print result.is_id

editticket

Edit a ticket, using an UPDATE statement on an existing ticket.

Definition and Parameters

editticket(xs:short is_id, xs:string reported_by, xs:string issue_type, xs:string reported_priority, xs:string description, xs:string product, xs:string release, xs:string keywords)

is_id: The ID # of the ticket.

Reported_by: The name of the person who is reporting the ticket. Must be in format: “<last_name>, <first_name>”, ie. “Johnson, Tamara”.

Issue_type: The type of issue, ie. “Problem”, “Question”, etc.

Reported_priority: The priority level that the customer is reporting for the ticket, ie. “Severe”.

Description: The main description of the ticket.

Product: The product pertaining to the ticket, ie. “ATMS Web”.

Release: The release pertaining to the ticket, ie. “5.10”.

Keywords: Any keywords associated with the ticket. Free form string.

Return

Returns a complex python object.

S_editticket.retc: Return code.

0: Success.

-1: Database connectivity error.

-2: Cannot find matching ct_id for “Reported By” name.

-3: Error during update.

Example

result = client.service.editticket(11049, "Fay, Matthew", "Problem", "Serious", "This is a ticket description", "ATMS Connect", "5.20", "These are keywords")

print result.retc

getidandcompany

Based on email address, get the associated contact/AQTS ID and their company name.

Definition and Parameters

getidandcompany(xs:string email)

email: The email address.

Return

Returns a complex python object.

S_getidandcompany.retc: Return code.

0: Success.

-1: Database connectivity error.

-2: Error during SELECT.

S_getidandcompany.ct_id: The contact/AQTS ID.

S.getidandcompany.company_name: The company name associated with the email address.

Example

result = client.service.getidandcompany(“customer@customer.com”)

print result.ct_id

insertuserforapproval

Insert user record to get approved/registered.

Definition and Parameters

insertuserforapproval(xs:string firstname, xs:string lastname, xs:string position, xs:string email, xs:string phone, xs:string password, xs:string company, xs:string optout)

firstname: User’s first name.

lastname: User’s last name.

position: User’s job position.

email: User’s e-mail.

phone: User’s phone.

Password: User’s password.

Company: User’s company name.

Optout: Either Y or N. Is user opting-out from AQT email notifications.

Return

Returns 0 for success, -1 for any error.

Example

result = client.service.insertuserforapproval("FirstName", "LastName", "Position", "Email", "Phone", "Pwd", "Company", "Y")

viewticketdetail

View all of the details of a specific ticket, including all of the responses.

Definition and Parameters

viewticketdetail(xs:decimal issue_id)

issue_id: The ID # of the ticket.

Return

Returns a complex python object.

S_viewticketdetail.retc: Return code.

0: Success.

-1: Database connectivity error.

-2: Error during retrieval process.

-3: Ticket not found.

-4: Error during retrieval process for getting ticket responses.

S_viewticketdetail.ticket_number: The ID # of the ticket.

S_viewticketdetail.reported_by: The name of the person who reported the ticket, in format "<last_name>, <first_name>".

S_viewticketdetail.reported_date: The date the ticket was reported, in format "mm-dd-yyyy".

S_viewticketdetail.assigned_to: Indicates where the ticket is currently assigned. Might be a person's name, ie. "Fay, Matthew" or a queue.

S_viewticketdetail.issue_type: The type of issue, ie. "Suggestion".

S_viewticketdetail.priority: The customer reported priority of the issue, ie. "Moderate".

S_viewticketdetail.product: The product associated with the ticket, ie. "ATMS Web".

S_viewticketdetail.release: Product release, ie. "5.00".

S_viewticketdetail.description: The main description of the ticket.

S_viewticketdetail.keywords: Any keywords associated with the ticket.

S_viewticketdetail.num_responses: The number of responses associated with the ticket.

S_viewticketdetail.response_date[]: A list of dates associated with each response.

S_viewticketdetail.response_name[]: A list of names/authors associated with each response.

S_viewticketdetail.response_text[]: A list of the actual text of each response.

Example

```
result = client.service.viewticketdetail(6488)

print result.retc

print result.num_responses

# Print first/oldest response.

print result.response_text[0][0]

# Print second response.

print result.response_text[0][1]

# When printing the response, there may be foreign Unicode
```

```
# characters in the string. For best results, encode unicode string as ascii,

# and ignore those foreign characters that can cause an error.

# http://stackoverflow.com/questions/3224268/python-unicode-encode-error?rq=1

test = (result.response_text[0][2]).encode('ascii', 'ignore')

print test
```

viewtickets

Return ticket summary data based on search criteria.

Definition and Parameters

viewtickets(xs:decimal issue_id, xs:string status, xs:string release, xs:string keywords, xs:decimal company_id)

issue_id: Ticket ID #. 0 for all.

Status: Status of the ticket, ie. “Open” or “Closed”. “0” for all.

Release: ie. “5.00”, “4.93”. “0” for all.

Keywords: Filter issue description based on keywords. For example, if the keywords argument is “setup”, only tickets with the word “setup” in the main description will come back. If the user doesn’t want to filter by keywords, use “0” as the argument for this parameter.

Company_id: ID # of the company associated with the person logged in. 0 for all.

Return

Returns a complex python object.

S_viewtickets.retc: Return code.

0: Success.

-1: Database connectivity error.

-2: Error during retrieval process.

S_viewtickets.ticket_number[]: A list of ticket numbers.

S_viewtickets.reported_by[]: A list of Reported By names, in the format “<last_name>, <first_name>”.

S_viewtickets.reported_date[]: A list of reported dates, in the format “mm-dd-yyyy”.

S_viewtickets.assigned_to[]: A list of Assigned To values, ie. “Hulegaard, Grant”.

S_viewtickets.issue_type[]: A list of issue type values, ie. “Question”.

S_viewtickets.priority[]: A list of the customer reported priority for each ticket.

S_viewtickets.product[]: A list of the products associated with each ticket.

S_viewtickets.release[]: A list of the release associated with each ticket.

S_viewtickets.short_description[]: A list of the short description associated with each ticket. Description truncated at 250 chars.

S_viewtickets.ticket_count: The number of tickets returned.

Example

```
result = client.service.viewtickets(0, '0', '0', '0', 2550)
```

```
print result.retc
print result.ticket_count
# Print first/newest ticket number and short description.
print result.ticket_number[0][0]
print result.short_description[0][0]
# Print next newest ticket number and short description.
print result.ticket_number[0][1]
print result.short_description[0][1]
```