3rd Party DBM Application Integration Manual

Cellocator Division
Pointer Telocation Ltd.

Proprietary and Confidential

Version 1.0

Revised and Updated: June 3, 2013

Legal Notices

IMPORTANT

1. All legal terms and safety and operating instructions should be read thoroughly before the product accompanying this document is installed and operated.
2. This document should be retained for future reference.
3. Attachments, accessories or peripheral devices not supplied or recommended in writing by Pointer Telocation Ltd. may be hazardous and/or may cause damage to the product and should not, in any circumstances, be used or combined with the product.

General

The product accompanying this document is not designated for and should not be used in life support appliances, devices, machines or other systems of any sort where any malfunction of the product can reasonably be expected to result in injury or death. Customers of Pointer Telocation Ltd. using, integrating, and/or selling the product for use in such applications do so at their own risk and agree to fully indemnify Pointer Telocation Ltd. for any resulting loss or damages.

Warranty Exceptions and Disclaimers

Pointer Telocation Ltd. shall bear no responsibility and shall have no obligation under the foregoing limited warranty for any damages resulting from normal wear and tear, the cost of obtaining substitute products, or any defect that is (i) discovered by purchaser during the warranty period but purchaser does not notify Pointer Telocation Ltd. until after the end of the warranty period, (ii) caused by any accident, force majeure, misuse, abuse, handling or testing, improper installation or unauthorized repair or modification of the product, (iii) caused by use of any software not supplied by Pointer Telocation Ltd., or by use of the product other than in accordance with its documentation, or (iv) the result of electrostatic discharge, electrical surge, fire, flood or similar causes. Unless otherwise provided in a written agreement between the purchaser and Pointer Telocation Ltd., the purchaser shall be solely responsible for the proper configuration, testing and verification of the product prior to deployment in the field.

POINTER TELOCATION LTD.'S SOLE RESPONSIBILITY AND PURCHASER'S SOLE REMEDY UNDER THIS LIMITED WARRANTY SHALL BE TO REPAIR OR REPLACE THE PRODUCT HARDWARE, SOFTWARE OR SOFTWARE MEDIA (OR IF REPAIR OR REPLACEMENT IS NOT POSSIBLE, OBTAIN A REFUND OF THE PURCHASE PRICE) AS PROVIDED ABOVE. POINTER TELOCATION LTD. EXPRESSLY DISCLAIMS ALL OTHER WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, SATISFACTORY PERFORMANCE AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL POINTER TELOCATION LTD. BE LIABLE FOR ANY INDIRECT, SPECIAL, EXEMPLARY, INCIDENTAL OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOSS OR INTERRUPTION OF USE, DATA, REVENUES OR PROFITS) RESULTING FROM A BREACH OF THIS WARRANTY OR BASED ON ANY OTHER LEGAL THEORY, EVEN IF POINTER TELOCATION LTD. HAS BEEN ADVISED OF THE POSSIBILITY OR LIKELIHOOD OF SUCH DAMAGES.

Intellectual Property

Table of Contents

1 Introduction

The purpose of this document is to define the mechanism and communication protocols in which an FM application provider's fleet management system (FMS) will interact with a 3rd party DBM application provider risk analysis system.

This document is divided into two main sections:

- The **interface protocol** which will be used by the two systems to transfer data in order to synchronize the two systems.
- The **SSO (Single Sign On) mechanism** that will enable the user to login once and move from one web application to another without the need to login to a second system.

1.1 Revision History

Version	Date	Description
1.0	02-06-2013	Edited

2 Web Services Specification

This section describes the web services exposed and used by the 3rd party DBM application provider and FM application provider systems.

The API used in this section is an example of a 3rd party DBM API; each 3rd party provider may have different structures, but with the same purpose.

The web services are divided into three usage categories:

- **Regular Behaviour:** This section contains all the web services which are required in order to keep the two systems in sync. Changes and updates which will be made on the FM application provider's FMS will normally require data synchronization between the FM application provider and the 3rd party DBM application provider and thus the relevant web services will be invoked by the FM application provider.

  To avoid data loss, these invocations must be queued upon server-level or network connection failure. The FM application provider's system will retry invoking the services until successful (or highlight the problem).
- **Data Integrity Verification:** Each system should have the ability to run comparisons on the data structure of the other system to insure data integrity. The protocol defines a set of web services that exposes the system structure. These web services will be exposed by each system, and will enable the other system to ensure data integrity.
- **Data Corruption Handling:** Once the data integrity check fails, there is a need to recover the system structure.

The following sections describe the three categories listed above.

2.1 Regular Behaviour

These methods are all implemented and exposed by the 3rd party DBM application provider and consumed by the FM application provider. In terms of Client/Server behavior, the 3rd party DBM application provider represents the Server and the FM application provider represents the Client.

In the below methods the 'changeType' field represents the type of change that the web method should perform in the 3rd party DBM application provider system.

Possible 'changeType' values are:

A: Add - a new record was added to the system

C: Change - a change was made to the record

D: Delete - record was deleted

Note that each account must have at least one sub account.

Signature	Description

`RESULT` **`AccountUpdate`**`(`
`string changeType (A/C/D),`
`string accountName,`
`int accountID,`
`DateTime creationDate,`
	Updates an account.
	Method-Specific return Values:
	▪ **ILLEGAL_ACCOUNTID** (3): The specified `accountID` does not exist.

Signature	Description
int timeZoneOffset, bool useDaylightSavings, string addressLine1, string addressLine2, string city, string state, string zip, string country, string phoneNumber, string extension, string faxNumber)	
RESULT **SubAccountUpdate**(string changeType (A/C/D), int subAccountID, int accountID, string subAccountName, DateTime time)	Updates a sub account associated with an account. Method-Specific Return Values: ▪ **ILLEGAL_ACCOUNTID** (3): The specified accountID does not exist. ▪ **INVALID_SUBACCOUNT** (4): The requested subAccountID does not exist (C/D) or already exists (A).
RESULT **VehicleSubAccountUpdate**(string changeType (A/D), int subAccountID, int vehicleID, DateTime time)	Updates a Vehicle-SubAccount Assignment. Method-Specific Return Values: ▪ **ILLEGAL_SUBACCOUNTID** (4): The requested subAccountID does not exist or exists under a different account. ▪ **ILLEGAL_VEHICLEID** (10): The requested vehicleID does not exist. Note: a Vehicle may belong to more than one SubAccount.
RESULT **VehicleUpdate**(string changeType (A/C/D), int vehicleID, int accountID, string imei, string plate, string vehicleAlias, string make, string model, VEHICLE_TYPE type, string vin, int timeZone, bool useDaylightSavings, DateTime time, bool isActive)	Updates a vehicle record. Method-Specific Return Values: ▪ **SUCCESS**: VehicleID and AccountID exist, vehicle is assigned to account and no other vehicle holds the same Plate, IMEI or VIN. ▪ **ILLEGAL_ACCOUNTID** (4): The requested account does not exist or vehicle belongs to a different account. ▪ **VEHICLEID_EXISTS** (9): VehicleID has already been assigned to another vehicle. ▪ **PLATE_EXISTS** (11): License plate has already been assigned to another vehicle. ▪ **IMEI_ALREADY_IN_USE** (12): IMEI already assigned to a different (active) vehicle.

Signature	Description
RESULT **DriverUpdate**(string changeType (A/C/D), int driverID, int accountID, string firstName, string middleName, string lastName bool isActive)	Updates a vehicle record. Method-Specific Return Values: ▪ **SUCCESS**: DriverID and AccountID exist, driver is assigned to account. ▪ **ILLEGAL_ACCOUNTID** (4): The requested account does not exist or driver belongs to a different account. ▪ **DRIVERID_EXISTS** (26): DriverID has already been assigned to another driver.
RESULT **VehicleDriverUpdate**(string changeType (A/D), int vehicleID, int driverID, DateTime time)	Updates a Vehicle-Driver Assignment. Method-Specific Return Values: ▪ **ILLEGAL_ DRIVERID** (27): The requested DriverID does not exist or exists under a different account. ▪ **ILLEGAL_VEHICLEID** (10): The requested VehicleID does not exist.
RESULT **UserAccountUpdate**(string changeType (A/D), int accountID, int userID, DateTime time)	Updates a User-Account Assignment. Method-Specific Return Values: ▪ **ILLEGAL_ACCOUNTID** (4): The requested accountID does not exist or exists under a different account. ▪ **ILLEGAL_USERID** (8): The requested userID does not exist. Note: a user may belong to more than one Account.
RESULT **UserSubAccountUpdate**(string changeType (A/D), int subAccountID, int userID, DateTime time)	Updates a User-SubAccount Assignment. Method-Specific Return Values: ▪ **ILLEGAL_SUBACCOUNTID** (4): The requested subAccountID does not exist or exists under a different account. ▪ **ILLEGAL_USERID** (8): The requested userID does not exist. Note: a user may belong to more than one SubAccount.
RESULT **UserDriverUpdate**(string changeType (A/D), int driverID, int userID, DateTime time)	Updates a User-Driver Assignment. Method-Specific Return Values: ▪ **ILLEGAL_ DRIVERID** (27): The requested DriverID does not exist or exists under a different account. ▪ **ILLEGAL_USERID** (8): The requested userID does not exist.

Signature	Description
bool **UsernameExists**(string username)	Checks the database for a User record with the specified username. Must be invoked by FM application provider on SAFETY APPLICATION prior for invoking the **UserUpdate(…)** Web Service Returns: True - User record already exists. False - There is no user record with that name.
RESULT **UserUpdate**(string changeType (A/C/D), int accountID, int userID, string userName, string SSOID, string firstName, string middleName, string lastName, bool isActive)	Updates a user record. When adding a user (changeType='A') SSOID must be supplied as an argument. Method-Specific Return Values: ▪ **SUCCESS**: The user was added or updated correctly (when adding, the name does not exist in the system). ▪ **ILLEGAL_ACCOUNTID** (3): The specified accountID does not exist. ▪ **USERNAME_EXISTS** (5): The username already exists. ▪ **USERID_EXISTS** (6): The requested userID already exists. ▪ **ILLEGAL_SSOID** (7): SSOID cannot be changed.
RESULT **UserEmailUpdate**(string changeType (A/C/D), int userID, int sequenceNumber, string emailAddress, bool useForReports)	Updates a user's email address. Method-Specific Return Values: ▪ **ILLEGAL_USERID** (8): The requested userID does not exist.
RESULT **UserPhoneUpdate**(string changeType(A/C/D), int userID, int sequenceNumber, string phoneNumber, string extension, PHONE_TYPE phoneType)	Updates a user's phone number. Method-Specific Return Values: ▪ **ILLEGAL_USERID** (8): The requested userID does not exist.

2.1.1 *Data Types*

2.1.1.1 RESULT

Enumeration is used as a common return value to indicate success, data problems or other errors. Additional values are added for each method.

Field name	Description
SUCCESS = 0	Web Service request completed successfully.
MISC_FAILURE = 1	There is an undefined error with the web service request.
RECORD_NOT_FOUND = 2	The record being updated or deleted could not be found.
DATA_CORRUPTION = 3-9999	Method-specific failure codes.
ILLEGAL_ACCOUNTID = 3	The specified account Id does not exist.
ILLEGAL_SUBACCOUNTID = 4	The requested SubAccountID does not exist or user/vehicle belongs to a different sub account.
USERNAME_EXISTS = 5	The username being added already exists.
USERID_EXISTS = 6	The requested userID already exists in one of the other systems.
ILLEGAL_SSO = 7	SSOID cannot be changed.
ILLEGAL_USERID = 8	The requested userID does not exist.
VEHICLEID_EXISTS = 9	VehicleID has already been assigned to another vehicle.
ILLEGAL_VEHICLEID = 10	The requested vehicleID does not exist.
PLATE_EXISTS = 11	License plate has already been assigned to another vehicle.
IMEI_ALREADY_IN_USE = 12	IMEI already assigned to a different (active) vehicle.
NO_SUCH_VEHICLE = 14	
NO_SUCH_ACCOUNT = 15	
VHCL_NOT_MEMBER_OF_ACCOUNT = 16	
ACCOUNT_EXISTS = 19	
CROSS_SUBACCOUNT_BINDING = 20	
BAD_PARAMETER = 23	
UNSUPPORTED_TIMEZONE = 24	

Field name	Description
ACCOUNT_NOT_EXISTS = 25	
DRIVERID_EXISTS = 26	DriverID has already been assigned to another driver.
ILLEGAL_DRIVERID = 27	The requested DriverID does not exist.
NET_ERROR = -1	

2.1.1.2 PHONE_TYPE

Enumeration is used to indicate the type of phone number.

Field	Description
BUSINESS = 0	Business phone number
FAX = 1	Fax phone number
PERSONAL = 2	Personal phone number
CELLULAR = 3	Cell phone

2.1.1.3 VEHICLE_TYPE

Enumeration is used to indicate the type of a vehicle.

Field	Description
NONE = 0	Value not set
PRIVATE_PASSENGER_AUTO = 1	
COMERCIAL_SIZE_AUTO = 2	
TRUCK = 3	
SEMI_TRAILER = 4	
BUS = 5	

2.2 Data Integrity Verification

This section provides a Full System structure comparison.

As the SAFETY APPLICATION structure is fully dependent on the correct invocation of web services by the FM application provider, there is a need to verify, on a daily basis, that all system entities defined in the FM application provider are also defined in the SAFETY APPLICATION system.

To support this comparison, each system should implement the web services in the following table.

Signature	Description
AccountIdentity[] **GetAccountList**()	Returns a list of all existing accounts.
AccountRecord **GetAccount**(int accountID, int options)	Returns the specified account's details. 'options' argument may contain values as described in the *Options Method Parameter* section.
SubAccountRecord **GetSubAccount**(int subAccountID, int options)	Returns the specified sub account's details. 'options' argument may contain values as described in the *Options Method Parameter* section.
DriverRecord **GetDriver**(int driverID, int options	Returns the specified driver's details.
VehicleRecord **GetVehicle**(int vehicleID, int options	Returns the specified vehicle's details.
UserRecord **GetUser**(int userId, int options)	Returns the user's details. 'options' argument may contain values as described in the *Options Method Parameter* section.
PhoneRecord **GetPhone**(int userId, int sequence)	Returns the user's phone details. 'sequence' represents the ID of the record sequence ID.
EmailRecord **GetEmail**(int userId, int sequence)	Returns the user's email address. 'sequence' represents the ID of the record sequence ID.

2.2.1 *Options Method Parameter*

The Options parameter is present only in: GetAccount(…), GetSubAccount(…), GetUser(…), GetDriver(…), and GetVehicle(…) web methods.

This argument enables the retrieving of the detained internal structure of the queried entities.

The following values represent the information retrieval categories:

- 0 = no additional information is returned
- 1 = account information is returned
- 2 = sub account information is returned
- 4 = vehicle information is returned
- 8 = driver information is returned
- 16 = users information is returned
- 32 = email information is returned
- 64 = phone information is returned

The 'options' argument can contain the above values and combinations of them.

128 will return the complete account structure.

For a complete comparison algorithm, see Data Corruption Handling.

2.2.2 *Data Types*

2.2.2.1 AccountIdentity

This type represents a single account's unique identification entity.

Field	Description
`string ID`	The unique account ID
`string Name`	The account's name

2.2.2.2 AccountRecord

This type represents an account and all its associated sub accounts.

Field	Description
`int AccountID`	The Account Unique ID
`string AccountName`	The Account name
`DateTime creationDate`	
`bool UseDaylightSavings`	Use day light savings or not
`string AddressLine1`	First address
`string AddressLine2`	Second address
`string City`	City

Field	Description
string State	State
string Zip	Zip code
string Country	Country
string PhoneNumber	Phone number
string Extension	Extension
string FaxNumber	Fax number
UserRecord AdminUSer	Account associated Admin User
SubAccountRecord[] SubAccounts	Array of associated sub accounts
UserRecord[] Users	Array of associated account users
DriverRecord[] Drivers	Array of associated drivers
VehicleRecord[] Vehicles	Array of associated vehicle

2.2.2.3 SubAccountRecord

This type represents a sub account associated to an account.

Field	Description
int SubAccountID	The sub account's unique ID
string Name	The sub account's name
DateTime Time	Modify date
VehicleRecord[] Vehicles	Array of sub account's vehicles
UserRecord[] AdminUsers	Array of associated sub account admin users

2.2.2.4 VehicleRecord

This type represents a vehicle entity.

Field	Description
int VehicleID	The unique vehicle ID
string UnitIMEI	
string Name	Vehicle's name
string Plate	Vehicle's plate
string VehiclePlateAlias	Vehicle's plate alias
string Make	Vehicle's make
string Model	Vehicle's model
int TimeZone	Time zone
bool UseDaylightSavings	Use day light savings or not
DateTime Time	Last modified date

Field	Description
VEHICLE_TYPE VehicleType	Type of vehicle
string Vin	Vehicle identification number
DriverRecord Driver	Vehicle associated driver

2.2.2.5 DriverRecord

This type represents a driver entity.

Field	Description
int DriverID	The unique driver ID
string FirstName	Driver's first name
string MiddleName	Driver's middle name
string LastName	Driver's last name
UserRecord User	Driver's associated user
VehicleRecord Vehicle	Driver's associated vehicle

2.2.2.6 UserRecord

This type represents a user entity.

Field	Description
int UserID	The user's unique ID
string UserName	The user's name
string SSOID	The SSOID associated with the user
string FirstName	The user's first name
string LastName	The user's last name
bool IsActive	Indicates if the vehicle is active
EmailRecord[] Emails	An array of user's emails
PhoneRecord[] Phones	An array of user's phones

2.2.2.7 PhoneRecord

This type represents a phone entity.

Field	Description
int SequenceNumber	The unique identifier of the record in FM application provider's system
string PhoneNumber	Phone number
string Extension	always empty
PHONE_TYPE PhoneType	always 0

2.2.2.8 EmailRecord

This type represents an email entity.

Field	Description
`string SequenceNumber`	The unique identifier of the record in FM application provider's system
`string EmailAddress`	Email address
`bool UseForNotification`	Indicates whether the user should be notified when sending reports

2.3 Data Corruption Handling

The flow below defines the recommended algorithm for recovering discrepancies:

1. Query all accounts identities on both systems and compare them by AccountID.
   a. Accounts found only on SAFETY APPLICATION should be deleted [AccountUpdate('d'..)]
   b. Accounts found only on FM application provider should be created on SAFETY APPLICATION [AccountUpdate('a')]
2. For each account that exists on both systems:
   a. Compare account attributes by ID (name, address etc.).
   b. Compare account users:
      - Users that exist only on SAFETY APPLICATION should be removed from the account [UserUpdate ('d')]
      - Users that exist only on FM application provider should be created on SAFETY APPLICATION [UserUpdate ('a')]
      - For each user that exists on both systems, compare email and phone number attributes. Create, change or delete SAFETY APPLICATION entities as necessary.
   c. Compare account sub accounts:
      - Sub Accounts that exist only on SAFETY APPLICATION should be removed from the account [SubAccountUpdate ('d')]
      - SubAccounts that exist only on FM application provider should be created on SAFETY APPLICATION [SubAccountUpdate ('a')]
      - For each sub account that exists on both systems, compare vehicles and associated users. Create, change or delete SAFETY APPLICATION entities as needed.
   d. Compare account drivers:
      - Drivers that exist only on SX should be removed from the account [DriverUpdate ('d')]
      - Drivers that exist only on FM application provider should be created on SAFETY APPLICATION [DriverUpdate ('a')]
      - For each driver that exists on both systems, compare user and associated vehicle. Create, change or delete SAFETY APPLICATION entities as needed.
   e. Compare account vehicles:
      - Vehicles that exist only on SAFETY APPLICATION should be removed from the account [VehicleUpdate ('d')]
      - Vehicles that exist only on FM application provider should be created on SAFETY APPLICATION [VehicleUpdate ('a')]

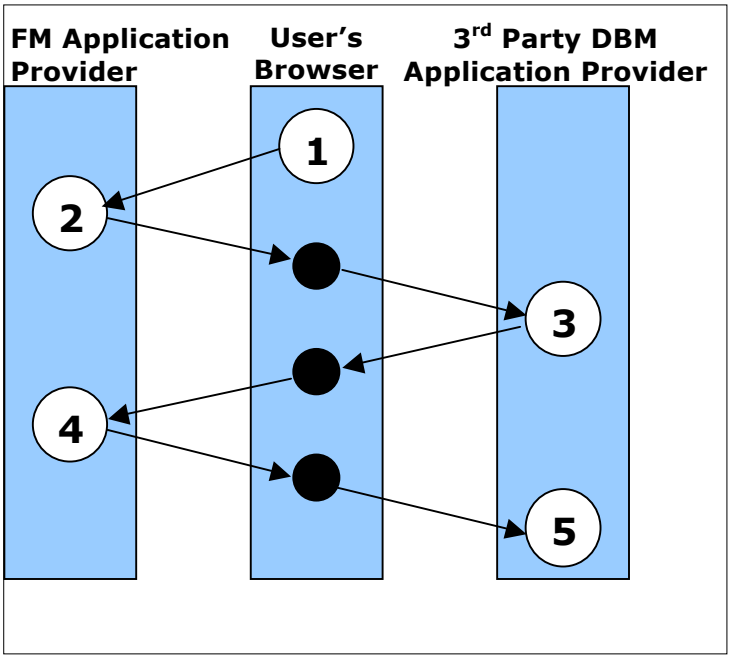
3 Single Sign-On (SSO)

SSO refers to the functionality that allows the FM application provider's customer portal users to be authenticated and logged in to the 3rd party DBM application provider's web application in a secured manner with only a single username/password login action via the FM application provider's website.

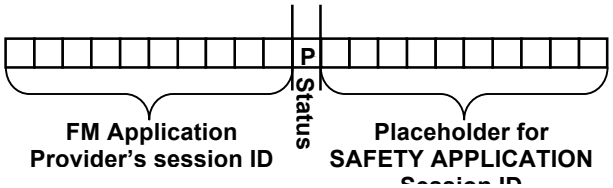
This mechanism simplifies user navigation between the FM application provider and 3rd party DBM application provider web applications and enhances the websites usability.

The 3rd party DBM application provider will provide a dedicated login page (e.g. SSOTransfer.aspx) that will handle the SSO login process. The FM application provider's customer portal will request this page (with HTTP GET) supplying encrypted query strings.

Flow diagram:

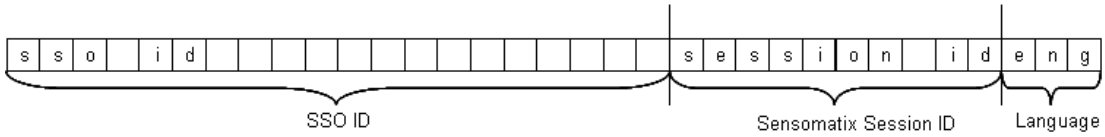


1. The user attempts to reach the 3rd party DBM application provider website by clicking the 3rd party DBM application provider link on the FM application provider's web application.
2. FM application provider generates a 10 digit unique ID for the user's browser session, sets a cookie on the browser for that active session only and formats the request string below:



The 'P' status marks the request as 'Pending'. Then the query string is encrypted using the Rijndael encryption algorithm and passed to the designated 3rd party DBM application provider login page [SSOTransger.aspx?login=<<Request>>).

3. The 3rd party DBM application provider's site retrieves the query string's "login" value from the URL in the SSOTransfer.aspx page, decodes and decrypts it and does the following:
   a. Verifies that the character at index 10 of the query string value is either ‘p’ or ‘P’ which means the request is pending.
   b. Verifies that the query string's length is 21 characters.
   c. Verifies that indexes 11 to 20 are empty.
   d. Creates an active session for the user, and sets a cookie (to pin the request to the browser/user session).
   e. Creates a new ID query string value by taking the first 10 characters of the passed in value, appending a status code of ‘A’ or ‘a’ (for 'Approved') and appending a 10 digit session ID after the status code. Then encrypts and URL encodes this new value into the query string and redirects back to the FM application provider (a page or http handler such as SSOTransfer.aspx?id=xxxxxxxxxxSyyyyyyyyyy).
4. The FM application provider retrieves the ID query string value in the SSOTransfer.aspx page and decodes and decrypts the query string, then does the following:
   a. Retrieves the original browser session ID value set in step 2 and verifies that the user/session and browser cookie are all valid.
   b. If all are valid, then a second request message for a SSO login is made to the 3rd party DBM application provider site containing the SSOID, 3rd party DBM application provider session ID, and language code.



5. The 3rd party DBM application provider decrypts the query string, validates that the string is valid and matches the part in the string that contains the 3rd party DBM application provider session ID against the current session ID.

   If the mentioned validations are successful, the username/password are retrieved through the mapping to the SSOID (obtained from the decrypted query string) and **the user shall login to the Safety application without the need to enter additional user/password.**