

040428MN-USF Data Objects Meeting 7 – Session Management

Agenda

- (1) Outline and understand the overall Session Management issue(s)
- (2) Discuss and understand each individual supplier position with regard to Session Management
- (3) Debate and agree the "technically correct" approach to Session Management (irrespective of individual supplier positions)
- (4) Debate, assess and agree impact on individual supplier positions if the "technically correct" solution were to be adopted
- (5) If possible, identify possible compromises with respect to the "technically correct" solution to accommodate individual supplier positions
- (6) Discuss and agree actions for individual suppliers to consider their positions with respect to the agreed "technically correct" solution or "technically correct with compromises" solution
- (7) AOB

Meeting Notes

- (1) Outline and understand the overall Session Management issue(s)**
- (2) Discuss and understand each individual supplier position with regard to Session Management**
- (3) Debate and agree the "technically correct" approach to Session Management (irrespective of individual supplier positions)**

End-Session

Discussion Points:

- Need a procedure to end a session to ensure server resources are not wasted
- Tenet have security and audit purposes – one person may only login once
- Tenet need the Manager to be able to end the session – Thales agree
- Need to be able to determine what sessions are running – which are dormant
- Therefore need a "global management session"
- Tenet would prefer to kill a session by "session name"
- Can we use "Session ID" or "User ID"
- The future may see multiple "common databases" and multiple "application servers" within one architecture – this needs to be considered

Provisionally agreed "technically correct" solution:

- An end of session function should be added -> Tenet and Siemens would prefer the "Manager" to handle the "End Session" using "Session ID" with "User ID" declared as optional for the purpose of "ending sessions" by a supervisor process rather than by UTMC applications (Tenet have implemented the "User ID" option)

Proposed IDL Code for agreement:

```
module SessionManager
```

```
{
```

```
    interface Manager
```

```
    {
```

```
        /*
```

```
        * frees server resources by ending the session created above
```

```
        * The Session parameter passed in identifies the session that will be ended and
```

```
        * is the value returned by CreateSession.
```

```
        */
```

```
        void endSession
```

```
        (
```

```
            in Session sess
```

```
        );
```

```
        /*
```

```
        * frees server resources by ending the session created above
```

```
        * This optional method is used by supervisory applications to forcibly terminate an existing
```

```
        * session identified by Name. Name parameter is that provided during the CreateSession.
```

```
        */
```

```
        void endSessionWithName
```

```
        (
```

```
            in String name
```

```
        );
```

Heartbeat

Discussion Points:

- All agree that this is a useful feature
- Requirement =
 - Client can determine if the Server has died
 - Server can determine if the Client has died

- Query Mechanism
- Subscription Mechanism
 - Push
 - Pull
 - Timeout
- Should Heartbeat be one-way or two-way ? ->
- Need to account for "clients" with long time periods of inactivity -> bollards and VMS
- Tenet would prefer not to use the Subscription Mechanism -> requires one Method to do two things which experience shows is less than ideal
- Alternative is a "Query Heartbeat" – Server interrogates Client or vice-versa
- Consideration needs to be given to applications written in languages in which these mechanisms are non-trivial to implement
- Should "Heartbeat" be mandatory ?
- If "Heartbeat" is mandatory then it is proposed that if a Client has not registered a "heartbeat" within a certain defined timescale it will be evicted
- Should there be a default timeout specified by the Client when it registers ("create session")? If no value is specified during the "create session" then the Server would adopt a default value
- Server to Client creates more redundant time for the Server while it waits for Client responses
- Should any activity from the Client be recognised as a "heartbeat" as well ? Is it necessary to have both "activity" and "heartbeat" ?

Provisionally Agreed:

- There is no requirement to have both Subscription and Query "heartbeat mechanisms"
- The preferred method is that the Client will invoke a "heartbeat" and register and maintain the "heartbeat" during its life (otherwise the Server will invoke default values potentially increasing the load on the Server)
- In addition any activity from the Client will be recognised by the Server as a "heartbeat" and will reset the Timeout
- Server-side implementation of the "Heartbeat" IDL is mandatory
- Client-side is not mandatory (but a default Timeout will be invoked by the Server and therefore Clients could be terminated by the Server as defined by the Server as above – Bullet 2)
- Agreed to reduce to a single session timeout parameter = "sessionTimeout" ("clientPollPeriod" will no longer be used)
- The "sessionTimeout" parameter is to be specified in Seconds (but care needs to be taken that undue load is not placed on the server)
- The proposed does put UTMC into conflict with the TIH "stay-alive" approach which is Server based – but this should not be an issue but will be discussed by Ian at the TIH Working Group on Protocols and Interfaces **A: IHC**

Proposed IDL Code for agreement:

```
module SessionManager
{
    interface Session
    {
        /*
        * Call made by a client so the server can monitor and timeout its connection
        * The parameter sessionTimeout shall be specified in Seconds and
        * shall be within a "default timeout period" range established by the Server
        * Returned value by Server is false if the parameter value is outside range else true.
        */
        boolean registerHeartbeat
        (
            in long sessionTimeout
        );

        /*
        * Called by the client to inform the Server that the Client is still alive.
        */
        void clientHeartbeat
        {
        };
    };
};
```

Binary Interface

Discussion Points:

- Issue arises through the need to overcome problems originating from CCTV – should this be resolved by other means – for example, passing CCTV images outside of the data object ?
- Thales have implemented the Data Object as defined – see "Provisionally Agreed"

Provisionally Agreed:

- Compromise agreed in the CCTV Data Object – stringify the image data and place in the database as part of the Data Object (large field)
- An option is to place the image data into the database as a file and point the Data Object at the file
- It was agreed that this method of working would be retained for now although there may be

need to re-visit this method of working in the future if a new application arises requiring the transfer of large binary files

Utility Interface

getTime

Discussion Items:

- Synchronisation by Clients of time with the CDB – latency ?
- Where do the “utilities” sit – new Utility Module ?
- Tenet need a “centralised time” for communication with on-street equipment
- If time is useful to other users then it should be in the Interface – otherwise it could sit outside
- Secondary function could be to identify any mis-timings between the CDB and on-street equipment
- The current method is to synchronise network systems via NTP (Network Time Protocol)

Provisionally Agreed:

- Agreed to slot into a Utility Module if it is agreed to introduce this Module in the future

getUniqueID

Discussion Items:

- Unique IDs are required to enable “objects” to be created and subsequently located
- “Get Unique ID”
- This will impact any data objects where Unique IDs are required
- Do we use the “Identity Column” ?
- A mechanism is needed for UTMC applications to track data object instances with unique IDs, for example VMS Commands

Provisionally Agreed:

- To defer until the Support Data Objects discussion within the Data Objects Sub-Group
- This will need to be resolved before release of Version 2

getRights

Discussion Items:

- Originally proposed to introduce a User Permissions Data Object ... but this duplicated functionality and information already available within databases
- It is proposed therefore that User Permissions are configured in the Database Interface

- Complicated by the need for “filters”
- Not clear if it will be possible to determine user rights by application – is it therefore better to use a Data Object Table -> User Permissions Data Object ?
- Need a generic method for “filtering” -> Server-side = what the Client is allowed to see -> Client-side = what the Client wants to see from the Server

Provisionally Agreed:

- To be reconsidered as part of determining a generic mechanism for filtering
- In the mean time the User Permissions Data Object will be considered as part of the Data Objects review discussions

Subscriptions for Pull Sequence

Discussion Items:

- Multiple results can already be achieved ? – only if mixed database actions are required would additional mechanisms be needed ?
- Order of results (when batched) could be an issue between different supplier implementations ?

Provisionally Agreed:

- That the IDL code below should be implemented (despite the issues surrounding ordering of results)
- Further individual consideration needs to be given to ordering of results before finalising the Proposed IDL Code description

Proposed IDL Code for final agreement:

```

module Subscriptions
{
    typedef sequence<DataNotification> DataNotifications;

    interface NotificationRequest
    {
        /*
        * This method is a more efficient version of the standard pull method
        * It returns all available data from multiple inserts/updates in one call
        * rather than the client making a call per updated data
        */
        DataNotifications pullSequence
        (
        );
    }
}

```

Delphi Name Clashes (raised by Brian Robinson)

Raised by Brian Robinson (Peek) in E-Mails "FW: UTMC IDL files" dated Mon 19/04/2004 10:54 and "USF : Session Management Issue" dated Thu 15/04/2004 10:37.

Release

Discussion Items:

- Proposed that "release" should be changed to "ReleaseObj" in the IDL code for the "Interface NotificationRequest"
- "ReleaseObj" not favoured -> "releaseObject" preferred

Provisionally Agreed:

The release function should not be used as there is already a release function defined in the COM object model.

Proposed IDL Code for agreement:

```
Interface NotificationRequest  
  
void releaseObject  
(  
);
```

Create

Discussion Items:

- Issue with the "sessionmanager.idl and subscriptions.idl files". These contain the Delphi reserved keyword "create" which causes a problem.
- Can this word be renamed to something else, or extended to reflect what the call is doing, for example "CreateSession" if it is creating a new session ?
- We also need to review other likely "clashes"

Provisionally Agreed:

- "create" in the Interface Session becomes "createObject"
- "create" in the Subscription Request becomes "createSubscription"
- "delete" in the Subscription Request becomes "deleteSubscription"

Other Name Clashes

enum DatabaseAction

Discussion Items:

- Enumerated Database Actions -> INSERT, DELETE, and UPDATE clash with some Microsoft

environments

Provisionally Agreed:

- Change -> INSERT, DELETE, and UPDATE to INSERT_ACTION, DELETE_ACTION, and UPDATE_ACTION

Interface Date-Time definition

Discussion Items:

- Julian Time has been used in a number of database implementations
- Issue was whether it had been implemented in the same way – the half-day clash (eg. With QMISS)
- Gareth to forward indicative information to Ian for discussion with the QMISS Team at MM.
A:GJ
- We need to agree on the definition and re-published as part of the updated UTMC specification

Provisionally Agreed:

- Janet to propose a “definition” for UTMC to be circulated and agreed by E-Mail **A: JA**

Data Types – Date-Time and Time on its own

-