POWER OF SIMPLICITY

Creating Compatible TDLs – Release 2.0 Onwards

Version: Creating Compatible TDLs-Release 2.0 Onwards/1.0/October 2010

Contents

Creating Compatible TDLs–Release 2.0 Onwards

Creating Compatible TDLs–Release 2.0 Onwards

Tally.ERP9 Release 2.0 bears remarkable changes in TDL in order to deliver Remote Edit Capability and other major enhancements in the form of File I/O capability,Timer Events,Variable persistence at Report Scope,Refresh Data and many more. In our constant effort to deliver the best technology ever and to move forward, we need to bring in lots of capability enhancements in the product from time to time. This involves changes to the language capability along with improvements in the performance.

Although we have tried to ensure maximum backward compatibility, there may be some cases where application developer may require to validate/rewrite the existing TDL codes to make them compatible with Tally.ERP9 Release 2.0.The following section completely focusses on the type of changes which may be required for converting the existing TDLs to be compatible from Release 2.0 onwards.This document will give you a fair idea on the level of changes required.

1. For Server Side Compatibility Only

1.1 Collection Attribute Fetch/Compute

Prior to Release 2.0, it was not necessary for the simple Collections to fetch or compute Methods, UDFs and/or Sub Collections at the Server. It was only needed if required for the Remote Client. From the breakthrough Release 2.0, all the required Methods and/or UDFs within a Collection are required to be fetched or computed before use even in the Server side in order to enhance the performance through good memory management by way of collecting only the required methods from each Object within the collection.

Syntax

```
[Collection: <Collection Name>]

   Type : <Required Object Type>
   Fetch: <Method1>[, <Method2>, <UDF>, <Sub Collection1>, ….]
```

Example:

```
[Collection: Sample List of Vouchers]
   Type : Voucher
   Fetch: Date, Voucher Number, Reference, Narration, InventoryEntries.*
```

The previous Collection gathers the methods Date, Voucher Number, Reference, Narration and Methods of the Sub Objects Inventory Entries from every Voucher.

Please note that it is a good practice to use summary collections and fetch only the required methods.

1.2 Default TDL Changes

Default TDL Codes are revamped to bring in better performance. Few definition names which might possibly be used in customized TDL codes are altered.

- Default TDL definitions using Simple TDL Collections earlier are now replaced with Extract Collections

  In customized codes, some default TDL Collections may have been modified to apply filters & other changes done but in Release 2.0, the same collection is replaced with extract collections in multiple places.

Example:

Collection/Table Active Batches which was used to display the Table in various Default TDL fields is now replaced with extract collections viz., Active Batches VchExtract.

```
[!Field: VCHBATCH OrdrName]
   Table : Active Batches VchExtract, New Number, Any
```

Hence, the filters applied in the Table Active Batches would not effect.

Solution:

The Solution for the above issue is to apply filters to the revised Collection.

```
[#Collection: Active Batches VchExtract]
    Add : Filter : Custom Filter
```

1.3 Voucher Creation

In Release 2.0, while creating an Voucher Object using User Defined Functions, a Variable SVViewName and a Method in the Voucher Object PersistedView needs to be set with the following predefined values for the nature of Voucher:

- AcctgVchView – For all Accounting Vouchers
- InvVchView – For all Invoicing Vouchers
- PaySlipVchView – For all Payroll Vouchers
- ConsVchView – For Stock Journal Vouchers

The above values must be converted into System Name using function SysName.

Example:

```
[Function: Create Sales Vch]
   00: SET : SVViewName: $$SysName:InvVchView
   05: NEW OBJECT: Voucher
   10: SET VALUE: Persisted View: ##SVViewName
        |
        |
        |
   90: CREATE TARGET
```

1.4 Multi Objects

Report Attribute Multi Objects is introduced to be used in case Multiple Objects of the same collection are being added or modified within the current Report.

Syntax

```
      [Report: <Report Name>]

          Multi Objects: <Edit Collection>
```

Where Edit Collection is the Collection Name in which Objects are being modified.

Example:

```
[Report: TSPL Smp Multi Objects]
   Multi Objects: TSPL Smp MO Vch Collection
```

The above Report TSPL Smp Multi Objects is designed to add/alter the methods of the Object within the Collection TSPL Smp MO Vch Collection.

2. For Client Side Compatibility Also

2.1 Attribute Fetch Object

When Multiple Methods of a single/multiple Objects are required, then that Object needs to be fetched at the Report/Form/Field or Function Definition wherever required. In other words, all the required methods are fetched from the Server. Object Identifier can be dynamically evaluated from an expression during run time.

Syntax

```
Fetch Object: <Object Type> : <Identifier Expression> : <List of Methods>
```

Example:

```
[Function: TSPL Smp LedParent]
   FETCH OBJECT: Ledger : ##pLedgerName : Parent, OpeningBalance
```

The Methods Parent and Opening Balance will be fetched for the selected ledger dynamically from within the variable pLedgerName under the Object Ledger.

Multiple Objects of the same Object Type can also be fetched using new Function Fetch Separator.

Example:1

```
[Function: TSPL Smp LedParent]
   FETCH OBJECT: Ledger : "Led1" + $$FetchSeparator +
                 "Led2" : Parent, OpeningBalance, ClosingBalance
```

The Methods Parent, Opening Balance and Closing Balance will be fetched for both the ledgers Led1 and Led2.

Example:2

```
[Report: List Var in Rep]
   Fetch Object: Ledger: @@TSPLSmpAllLedgerList : LedListVar.*

[System: Formula]
   TSPL Smp AllLedgerList : $$FullListEx:$$FetchSeparator:Ledger:$Name
```

In the above Report, the UDF LedListVar is being fetched for all the Ledgers.

2.2 Collection Attribute ParmVar

Collection Attribute ParmVar is introduced in order to evaluate the desired expression at the Client end itself where the current Object context is available and subsequently transport the same to the Server.

In Remote Client environment where a Collection needs to be gathered with various Filters and/or Child of condition involving requestor object context, the remote server is unaware of the current object context at the Client and so such Methods cannot be evaluated during run time. Hence,

Collection Attribute, Parm Var evaluates the expressions at the Client end and transports the variable value to the Server.

Syntax

```
[Collection: <Collection Name>]
    Child Of: ##<Pam Var Name>
    Parm Var: <Parm Var Name>: String : <Formula>
```

Example:

```
[Collection: List of Ledgers under Selected Group]
   Type     : Ledger
   ChildOf  : ##ParmVarGrp
   Parm Var: ParmVarGrp : String : $Name
```

The Collection List of Ledgers under Selected Group evaluates the Method Name in the Remote Client end and the same is passed to the server through Parameter Variable ParmVarGrp.

2.3 Report Attribute Fetch Values

Multiple Methods of the current Object can be fetched using Report Attribute Fetch Values. In other words, when methods are to be evaluated in current object context, Fetch Values must be used with the list of methods required.

Syntax

```
[Report: <Report Name>]
   Fetch Values: <List of Methods>
```

Example:

```
[Report: Report Object Association]
   Object       : Stock Item : ##SStockItem
   Fetch Values: Opening Balance, Closing Balance, Parent, Description
```

The Methods Opening Balance, Closing Balance, Parent and Description will be fetched for the current Stock Item Object context set in the Report Definition.

2.4 Collection Attribute Client Only

Collection Attribute Client Only must be enabled if the Collection needs to be fetched from the client itself and the Request need not go to the Server.

Example:

```
[Collection: Data Source Report Selected]
   Data Source : Report : Selected Lines
   Client Only : Yes
```

The above Collection evaluates at the Client itself without sending a request to the Server.

Whenever the Data needs to be gathered from external Data Sources like XML, DLL, ODBC Collection. In those cases the Client Only Attribute needs to be set to “Yes” as we do not need the request to be sent to the Server.We will take a few examples to depict this as below.

Also DLL Collection where the collection needs to be gathered at the Client end needs to enable Client Only.

Example:

```
[Collection: DLL XML Coll with Inp Parameter]
   Data Source     : AxPlugin XML : Sample4.Class1
   XML Object Path : Result
   Input Parameter : ##DLLInputParameter
   Client Only     : Yes
```

The above DLL Collection evaluates at the Client itself. Similarly, Collection with Compound/ Simple List Variables needs to be gathered from within the Client itself hence Client Only attribute must be enabled.

Example:

```
[Collection: LV List Collection]
   Data Source: Variable: LVEmp
   Format     : $LVEmp
   Format     : $$VarKey
   ClientOnly : Yes
```

The above Collection gathered using Variable as its Data Source is evaluated at the Client itself.

2.5 Action Fetch Object Info

Action Fetch Object Info is introduced to fetch object related information on demand. This action requires to be triggered from an existing Object Context. For Example, when a user explodes a

Daybook Report to see the details of the current voucher, Action Fetch Object Info sends a request to fetch the current Object information.

Syntax

Fetch Object Info: <List of Methods> : [<Object Type> : <Identifier>]

Example:

```
[!Key: Fetch AccInfo]
   Action : Fetch Object Info : PartyLedgerName, PartyName, Reference, +
        AllLedgerEntries.*,AllLedgerEntries.StockAffects, +
        AllLedgerEntries.IsBillWiseOn,AllLedgerEntries.IsCostCentresOn  +
        : Voucher : @@CurrVchKey

[System: Formula]
   CurrVchKey : $$SPrintf:"ID:%s":$MasterID
```

The above Key Option is used in Voucher Register where on pressing Shift + Enter or Exploding a Line, details pertaining to the current voucher is displayed below.

This is done to enhance the performance of the Report at the Client end as the detailed information pertaining to all the Line Items (Objects) are not needed to be fetched thereby occupying the memory when the user may choose to dig into only one object detail.