ODPP v1.0 to v2.1.0.3
Migration Guide

IBM's Optim Enterprise Solution

Version 9 Release 1 Modification 0 Fixpack 3 (March 2013)
This edition applies to version 9, release 1, modification 0, fixpack 3 of IBM InfoSphere Optim Masking on Demand and to all subsequent releases and modifications until otherwise indicated in new editions.

1. Introduction

This guide contains the details for migrating an application using the ODPP Application Programming Interface (API) from v1.0 to v2.1.0.3

There are some significant changes to the structures for ODPP v2.1.0.3 - especially for supporting single-byte character set (SBCS)/multi-byte character set (MBCS) strings.

2. Common changes

This section details the changes that are common to the new ODPP v2.1.0.3 API.

1. Use INITIALIZE_ODPP_STRUCT or INITIALIZE_ODPP_STRUCT_PTR for all structure-type objects. This is required for proper versioning of structures and should not be omitted.
2. Replace all the old structure instances with new structure instances as explained below in the following sections.
3. All fixed length buffers are now replaced by pointers which need to be allocated by the calling application. The number of bytes allocated to individual buffers should be set in the buffer length members of the structures.
4. All column names, parameter names and parameter values no longer need to be NULL terminated.
5. The header file ODPPCmnAPI10.h has been renamed to ODPPCmnAPI.h
6. The header file ODPPCmnErrorCodes10.h has been renamed to ODPPCmnErrorCodes.h
7. The library ODPProvider10.dll has been renamed to ODPProvider2.1.0.3.dll for Windows. The libODPProvider10.so has been renamed to:
   libODPProvider.sl.2.1.0.3 for HP-UX
   libODPProvider.2.1.0.3.a for AIX
   libODPProvider.so.2.1.0.3 for other UNIX/Linux-type systems.

3. Function: Provider_FrmwInit

1. Argument pInitParams, a pointer to structure DP_INIT_OP_DEF is replaced by pointer to a new structure DP_FRMW_PARAMS_DEF.
2. Argument sOprCount is removed from the function argument list.

3.1 For ODPP v1.0 users

1. If you are currently passing NULL for the 1st argument of Provider_FrmwInit then continue to pass NULL in this new version.
2. If you are providing input via DP_INIT_OP_DEF then follow these steps:
   - Change to the new version of DP_INIT_OP_DEF as explained in the section ***Structure: DP_INIT_OP_DEF***
   - Create an object of DP_FRMW_PARAMS_DEF and set the pParams member with a pointer to the new object of ***DP_INIT_OP_DEF*** created in the previous step.

4. Function: Provider_Init

1. A new function argument is introduced as the 3rd parameter, iSVCIdLen. This supplies the length, in bytes, of the character string pointed to by the 2nd parameter pSvcId.

2. DP_SVC_DEF structure is now updated and uses newer versions of DP_FIELD_DEF and DP_INIT_OP_DEF

5. Structure: DP_FIELD_DEF

1. Member sWCHAR has been removed. It is no longer needed as the union COLNAME_SS has separate pointers for a wide-character-type and mixed-character-type sub-structures that contain not only the column name but also the DBMSType and code page.
2. iCCSID has been changed to iDataCodePage
3. cDataDBMSType has been added. It identifies the DBMS origin of the data when the data originates in a DBMS. This is important for the ODPP handling of DBMS-specific code pages.
4. iColNameBytes has been added to specify the number of bytes in the column name field pColName within the DPFD_WC_SS or DPFD_MC_SS sub-structures.
5. ColName field has been replaced by the DPFD_MC_SS (for SBCS/MBCS) DPFD_WC_SS (for Unicode/wide-character) structures using the union CN.

5.1 For ODPP v1.0 users

For each DP_FIELD_DEF the following changes are applicable for supplying the column name.

1. Allocate an object of DPFD_WC_SS and assign it to <field def object>.CN.pWC
2. Allocate the required memory for Column Name and assign it to <field def obj>.CN.pWC->pColName buffer
3. Set <field def obj>.iColNameBytes to the size of pColName buffer in bytes (including null terminator if any)

6. Structure: DP_INIT_OP_DEF

1. usOperand has been changed to usParameterID to specify the ODPP parameter identifier.
2. Structure OPVAL has been replaced by the union PV to specify parameter values in either Unicode/wide-character format for mixed character (SBCS/MBCS) format.
3. pVal has been replaced by structure DPPRM_VAL_WC_SS using union PV.
4. Added support to pass parameters by name in addition to the existing parameter identifier with the addition of the following:
   - cParamNameSubType specifies the type of data contained in the iParamNameBufBytes.
   - iParamNameBufBytes contains the parameter name.
5. Added support for SBCS/MBCS and User Defined values with the following:
   - pMC supports passing SBCS/MBCS values.
   - pUserVal supports passing user defined values as a pointer.
6. iValueSubType added to specify type of value being passed for a given ODPP parameter

6.1 For ODPP v1.0 users

1. Change all instances of usOperand to usParameterID.
2. Set iValueSubType to any one of the following which best describes the value:
   PARAM_VAL_NONE 0 parameter has no value
   PARAM_VAL_NUM 1 parameter value is numeric
   PARAM_VAL_WC 2 parameter value is wide-character
3. If value is PARAM_VAL_NUM change all instances of <param obj>.OPVAL.uiVal to PV.uiVal
4. If value is PARAM_VAL_WC allocate an object of DPPRM_VAL_WC_SS and assign it to <param obj>.PV.pWC.
5. Allocate the required memory for parameter wide char value and assign it to <param obj>.PV.pWC->pParamVal
6. Replace all instances of <param obj>.OPVAL.pVal with <param obj>.PV.pWC->pParamVal
7. Set <param obj>.iValueBufBytes to size of pParamVal in bytes (including NULL terminator if any)

7. Structure: DP_SVC_DEF

1. sOprCount has been renamed to sParamCount.
2. pOperands has been renamed to pParams.
3. Added support for passing a global code page value for column names via the structure DP_FIELD_DEF pFldDef and the structure DP_INIT_OP_DEF pParams that contain the parameter names and values.

7.1 For ODPP v1.0 users

1. Change all instances of sOprCount to sParamCount.
2. Change all instances of pOperands to pParams.

8. Function: Provider_Service

1. No changes to the arguments for this function.
2. Use newer versions of DP_SVC_DEF and DP_ROWSET_DEF.

9. Structure: DP_ROWSET_DEF

1. No changes to the structure.

10. Structure: DP_ROW_DEF

1. No changes to the structure.

11. Function: Provider_Enumerate

1. No changes to the arguments for this function.
2. Use newer versions of DP_PRV_DEF.

12. Structure: DP_PRV_DEF

1. In the newer version this structure is broken into two parts:
   - DP_PRV_DEF
   - DP_PRV_DETAILS.
2. DP_PRV_DEF provides the global type information relating to the linked list of provider-type information including the related DBMSType and code page.
3. DP_PRV_DETAILS provides a link list of all the enumerated service providers. Detailed information is returned via the PD union of information being returned in either Unicode/wide-characters or mixed character (SBCS/MBCS).

12.1 For ODPP v1.0 users

For DP_PRV_DEF:

1. Allocate an array or a link list of DP_PRV_DETAILS equal to the number of providers expected.
2. Set pPrvDetails member pointer of DP_PRV_DEF structure to the array/linked-list allocated in step 1 above.

For DP_PRV_DEF:

1. **DO NOT USE INITIALIZE_ODPP_STRUCT or INITIALIZE_ODPP_STRUCT_PTR for DP_PRV_DETAILS**
2. Allocate an object of DPPD_WC_SS structure and set <DP_PRV_DETAILS obj>.PD.pWC with this address.
3. Allocate enough memory for Provider Name (Max being ODPP_PROVIDER_NAME_LEN wide characters) and set <DP_PRV_DETAILS obj>.PD.pWC->pProviderName with this address.
4. Set iPrvNameBufBytes to the size of pProviderName in bytes (including NULL terminator if any).
5. Allocate enough memory for Provider Details (Max being ODPP_PROVIDER_DESCRIPTION_LEN wide characters) set <DP_PRV_DETAILS obj>.PD.pWC->pDescription with this address.
6. Set iPrvDescBufBytes to the size of pDescription in bytes (including NULL terminator if any).

13. Function: Provider_GetError

1. No changes to the arguments for this function.
2. Use newer versions of DP_ECB.

14. Structure: DP_ECB

1. Change NumTokens to sNumTokens.
2. Change ErrCode to iErrCode.
3. Change lRowNum to iRowNum. The datatype changed from a ‘long’ to an ‘int’.

15. Function: Provider_GetFormattedErrorMsg

1. All arguments are now replaced by structure DP_ERR_MSG.

15.1 For ODPP v1.0 users

1. Allocate an object of DP_ERR_MSG.
2. If you already have a valid DP_ECB object from a Provider_GetError, then ignore this step and jump to step 3. If you do not have a valid DP_ECB object, set the pECB member to NULL and for the error where you want the details, set iErrNum to the error code and jump to step 4.
3. Set pECB to pointer of DP_ECB retrieved from Provider_GetError.
4. Allocate object of DPEM_WC_SS and set <DP_ERR_MSG obj>.EM.pWC with this address.
5. Allocate buffer for Error Message Source (Max is ODPP_MAX_ERR_MSG_SRC_LENGTH) and set <DP_ERR_MSG obj>.EM.pWC->pMsgSrc with this address.
6. Set <DP_ERR_MSG obj>.EM.pWC->iMsgSrcBytes to size of pMsgSrc in bytes (including NULL terminator if any).
7. Allocate buffer for Error Message (Max is ODPP_MAX_ERR_MSG_TEXT_LENGTH) and set <DP_ERR_MSG obj>.EM.pWC->pMsg with this address.
8. Set <DP_ERR_MSG obj>.EM.pWC->iMsgBytes to size of pMsg in bytes (including NULL terminator if any).
9. Allocate buffer for Error Message Body (Max is ODPP_MAX_ERR_MSG_BODY_LENGTH) and set <DP_ERR_MSG obj>.EM.pWC->pMsgBody with this address.
10. Set <DP_ERR_MSG obj>.EM.pWC->iMsgBodyBytes to size of pMsgBody in bytes (including NULL terminator if any).

16. ODPP v1.0 API and Structures

16.1 v1.0 API

This section provides a complete listing of the ODPP v1.0 API prototypes before the ODPP v2.1.0.3 changes:

```
RETVAL Provider_FrmwInit(DP_INIT_OP_DEF *pInitOperands,
                         short sOprCount);

RETVAL Provider_Init(int *pSvcToken,
                     char *pSvcID,
                     DP_SVC_DEF *pSvcdef,
                     char bAllocInitBlock);

RETVAL Provider_Service(int iSvcToken,
                     short sMethod,
                     DP_SVC_DEF *pSvcDef,
                     DP_ROWSET_DEF *pRowSet);

RETVAL Provider_Term(int iSvcToken);

RETVAL Provider_Enumerate(DP_PRV_DEF *pPrvDetails,
                        short *pCount);

RETVAL Provider_GetErrorCount(int iSvcToken,
                              short * pErrCount,
                              int * pErrAreaLen);

RETVAL Provider_GetError(int iSvcToken,
                         DP_ECB * pECB);

RETVAL Provider_GetFormattedErrorMsg(DP_ECB * pECB,
                                     int iLanguage,
                                     ODPP_WCHAR *pMsgSrc,
                                     ODPP_WCHAR *pMsg,
                                     ODPP_WCHAR *pMsgBody);

RETVAL Provider_TermErrors(int iSvcToken);

RETVAL Provider_GetInfo(int iSvcToken,
                        short sRequest,
                        void *ptr,
                        int *pBufLen);

RETVAL Provider_FrmwTerm(void);
```

16.2 v1.0 Structures

This section contains a complete listing of the ODPP v1.0 structures before the ODPP v2.1.0.3 changes.

16.2.1 DP_FIELD_DATA_DEF

```
/*-------------------------------------------------------------*
 * Name:          DP_FIELD_DATA_DEF                            *
 * Description:                                                *
 * A Field data that Definition will carry the source &        *
 * target data                                                 *
 *-------------------------------------------------------------*/
typedef struct s_DpField_Data_Define
{
   char           bSrcNull;   // Source Null indicator
   char           bDstNull;   // Dest Null indicator
   int            iSrcBufLen; // length, or max length of source buffer
                              // in bytes. On Return in case bCopyToDest of SvcDef
                              // is FALSE this variable will be updated to
                              // contain the size of the output data in bytes
   int            iDstBufLen; // length, or max length of dest buffer
                              // in bytes.  On Return in case bCopyToDest of SvcDef
                              // is TRUE this variable will be updated to
                              // contain the size of the output data in bytes
   RETVAL         iErrorCode; // Error code for this data
   unsigned char  *pSrcBuf;   // buffer to source field
   unsigned char  *pDstBuf;   // buffer to destination field
   struct s_DpField_Data_Define
                  *pNext;     // Next element in chain of DP_FIELD_DATA_DEF
} DP_FIELD_DATA_DEF;
```

16.2.2 DP_ROW_DEF

```
/*-------------------------------------------------------------*
 * Name:          DP_ROW_DEF                                   *
 * Description:                                                *
 * The row definition incapsulates the DP_FIELD_DEFINE struct  *
 * to provide a 2 dimensional array of DP_FIELD_DEFINE         *
  ********************IMPORTANT*********************************
   bProcessed should be FALSE (i.e. 0) for each row in first
   Service Call.
 ********************IMPORTANT*********************************
 *-------------------------------------------------------------*/
typedef struct s_dprow_def
{
   unsigned char        EyeCatcher[4]; // eye catcher
   char                 bProcessed;    // TRUE if the ROW processed earlier
                                       // This should be FALSE (i.e. 0) for each
                                       // row in first Service Call.
   char                 bHasError;     // TRUE if any column(s) has an error code
   short                sStructVer;    // version of this structure
   short                sStructLen;    // length of this structure
   short                sCount;        // Count of DP_FIELD_DATA_DEF(s) either
                                       // in form of an array or a chain of
                                       // elements
   DP_FIELD_DATA_DEF   *pFldDataDefine;// Pointer to an array of
                                       // DP_FIELD_DATA_DEF(s) OR pointer
                                       // to the first element of chain of
                                       // DP_FIELD_DATA_DEF.
   struct s_dprow_def   *pNext;       // Next element in chain of DP_ROW_DEF
}DP_ROW_DEF;
```

16.2.3 DP_ROWSET_DEF

```
/*-------------------------------------------------------------*
 * Name:          DP_ROWSET_DEF                                *
 * Description:                                                *
 * The rowset encapsulates a batch of DP_ROW_DEF               *
*-------------------------------------------------------------*/
#define  DPRST_ID     "DPRS"  /* Use specific definition.      */
#define  DPROW_VER1   1       /* DP_DEF_DEFINE Version 1       */
typedef struct s_dprowset_def
{
   unsigned char  EyeCatcher[4]; // eye catcher
   short          sStructVer;    // version of this structure
   short          sStructLen;    // length of this structure
   int            iCount;        // Count of DP_ROW_DEF(s) either in form of
                                 // an array or a chain of elements
   int            iLastPrcsRow;  // Index of the last row processed. Needed for
                                 // restart after discard limit failure.
                                 // This should be 0 for first Service Call.
   DP_ROW_DEF     *pRowDefine;   // Pointer to an array of DP_ROW_DEF(s)
                                // OR pointer to the first element of chain of
                                // DP_ROW_DEF.
   char           reserved[100];
}DP_ROWSET_DEF;
```

16.2.4 DP_FIELD_DEF

```
/*-------------------------------------------------------------*
 * Name:          DP_FIELD_DEF                                 *
 * Description:                                                *
 * A Field Definition should describe all of the attributes of  *
 * a field.  A field descriptor used for input will describe    *
 * all of the attributes of that input as well as setting a     *
 * pointer to that field.  A field descriptor used for output   *
 * will set a pointer to where the data should be placed, and   *
 * attributes should be set as to what form the data is in      *
 * that the caller expects. Masked data can be returned either  *
 * by modifying the Source or by the Target                     *
 *-------------------------------------------------------------*/
typedef struct s_DpFieldDefine
{
   unsigned char     EyeCatcher[4]; // eye catcher
   short             sStructVer;    // version of this structure
   short             sStructLen;    // length of this structure
   short             sWCHAR;        // Size of ODPP_WCHAR if different from current
                                    // platform. [Not used as of now]
   short             sDatatype;     // data type of the input data
   int               iLength;       // column length in case of string data types
   int               iPrecision;    // precision of the input decimal data,
                                    // and precesion of ORA VARNUM
   short             sScale;        // scale of the input decimal data
   int               iCCSID;        // field CCSID
   void              *ptr;          // pointer for internal use
   ODPP_WCHAR        ColName[MAX_COLNAME_SIZE + 1];   // Column Name.
   struct s_DpFieldDefine
                     *pNext;        // Next element in chain of DP_FIELD_DEFINE
} DP_FIELD_DEF;
```

16.2.5 DP_INIT_OP_DEF

```
/*-------------------------------------------------------------*
 * Name:          DP_INIT_OP_DEF                               *
 * Description:                                                *
 * Init Operand Definition contains all the initialization     *
 * operands required for the current service                   *
 *-------------------------------------------------------------*/
typedef struct s_init_op_def
{
   unsigned short    usOperand;  //Operand
   struct
   {
      unsigned int   uiVal;      //To supply numeric value
      ODPP_WCHAR     *pVal;      //To supply a wide string param value
   }OPVAL;
   struct s_init_op_def
                     *pNext;     //Pointer to next element in
                                 //chain of operands
}DP_INIT_OP_DEF;
```

16.2.6 DP_SVC_DEF

```
/*-------------------------------------------------------------*
 * Name:          DP_SVC_DEF                                   *
 * Description:                                                *
 * Init Definition contains the operands list along with other *
 * initialization parameters                                   *
 *-------------------------------------------------------------*/
#define        DPSVC_ID     "DPSV"  /* Use specific definition. */
#define        DPSVC_VER1   1       /* DP_DEF_DEFINE Version 1  */
typedef struct s_svc_def
{
   unsigned char  EyeCatcher[4]; // eye catcher
   char           bRefreshSVCDef;// TRUE to refresh the existing svc def
                                 // with this one  [Not used as of now]
   char           bCopyToDest;   // Copy the masked data to destination buf
   short          StructVer;     // version of this structure
   short          StructLen;     // length of this structure
   short          sOprCount;     // No. of elements in pOperands
   short          sFldCount;     // no of elements in pFldDef
   DP_INIT_OP_DEF *pOperands;    // pointer to Array or First element in
                                 // chain of INIT_OP. Providing all the
                                 // operands needed.
   DP_FIELD_DEF   *pFldDef;      // Pointer to an array of DP_FIELD_DEFINE(s)
                                 // or pointer to first element in the chain
                                 // of DP_FIELD_DEFINE(s)
   char           reserved[100];
}DP_SVC_DEF;
```

16.2.7 DP_PRV_DEF

```
/*-------------------------------------------------------------*
 * Name:          DP_PRV_DEF                                   *
 * Description:                                                *
 *    Contains the Provider details. Typically supplied by the *
 *    provider during call to enumerate the providers          *
 *-------------------------------------------------------------*/
#define        DPPRV_ID     "DPPV"  /* Use specific definition. */
#define        DPPRV_VER1   1       /* DP_DEF_DEFINE Version 1  */
typedef struct s_DpPrv_def
{
   unsigned char     EyeCatcher[4]; // eye catcher
   char              bUserDefine;   // TRUE if user supplied DP
                                    // Provider
   char              bLicensed;     // TRUE if licensed to use this DP
                                    // provider
   short             sStructVer;    // version of this structure
   short             StructLen;     // length of this structure
   short             sProviderID;   // internally defined provider ID
   short             sProviderVer;  // Version of the provider
                                    // internally defined provider Name
   ODPP_WCHAR        pProviderName[ODPP_PROVIDERNAME_BUF + 1];
                                    // provider description
   ODPP_WCHAR        pDescription[ODPP_PROVIDER_DESCRIPTION + 1];
   struct s_DpPrv_def
                     *pNext;         // Pointer to next element in chain
   char              reserved[100];
}DP_PRV_DEF;
```

16.2.8 DP_ECB

```
/*-------------------------------------------------------------*
 * Name:          DP_ECB                                       *
 * Description:                                                *
  * Error Control Block                                       *
 * It holds the error code, the row number where the error     *
 * occurred and optionally any number of tokens containing     *
 * specific details                                            *
 *-------------------------------------------------------------*/
typedef struct s_DP_ECB
{
   short             NumTokens;  /* Number of tokens           */
   int               ErrCode;    /* Error Code                 */
   long              lRowNum;    /* input row                  */
   struct s_DP_ECB   *pNext;     /* pointer to next ECB        */
   DP_TOKEN          aToken[1];  /* Error tokens               */
} DP_ECB;
```

17. ODPP v2.1.0.3 API and Structures

17.1 v2.1.0.3 API

```
RETVAL Provider_FrmwInit
                            (
                               DP_FRMW_PARAMS_DEF *pInitParams
                            );

RETVAL Provider_Init
                            (
                               int *pSvcToken,
                               char *pSvcID,
                               int iSvcIdLen,
                               DP_SVC_DEF *pSvcdef,
                               char bAllocInitBlock
                            );

RETVAL Provider_Service
                            (
                               int iSvcToken,
                               short sMethod,
                               DP_SVC_DEF *pSvcDef,       //FOR FUTURE USE
                               DP_ROWSET_DEF *pRowSet
                            );

RETVAL Provider_Term
                            (
                               int iSvcToken
                            );

RETVAL Provider_Enumerate
                            (
                               DP_PRV_DEF *pPrvDetails,
                               short *pCount
                            );

RETVAL Provider_GetErrorCount
                            (
                               int iSvcToken,
                               short * pErrCount,
                               int * pErrAreaLen
                            );

RETVAL Provider_GetError
                            (
                               int iSvcToken,
                               DP_ECB * pECB
                            );

RETVAL Provider_GetFormattedErrorMsg
                            (
                               DP_ERR_MSG *pErrMsg
                            );

RETVAL Provider_TermErrors
                            (
                               int iSvcToken
                            );

RETVAL Provider_GetInfo
                            (
                               int iSvcToken,
                               short sRequest,
                               void *ptr,
                               int *pBufLen
                            );

RETVAL Provider_FrmwTerm
                            (
                               void
                            );
```

17.2 v2.1.0.3 Structures

17.2.1 DP_ECB

```
//------------------------------------------------------------------------
// Name:          DP_ECB
// Description:
// Error Control Block (ECB) holds the error code,
// the row number where the error occurred
// and optionally any number of tokens containing specific details
//------------------------------------------------------------------------
typedef struct DP_ECB
{
   struct DP_ECB *pNext;               // Pointer to next ECB
   char           cEyeCatcher[4];      // Eye catcher
   short          sStructVer;          // Version of this structure
   short          sStructLen;          // Length of this structure
   short          sNumTokens;          // Number of tokens
   int            iErrCode;            // Error Code
   int            iRowNum;             // Input row
   DP_TOKEN       aToken[1];           // Error tokens
} DP_ECB;
```

17.2.2 DP_FIELD_DATA_DEF

```
//------------------------------------------------------------------------
// Name:          DP_FIELD_DATA_DEF
// Description:
// The Field Data Definition carries the source and target data
//------------------------------------------------------------------------
typedef struct DP_FIELD_DATA_DEF
{
   struct DP_FIELD_DATA_DEF
                 *pNext;               // Next element in chain of DP_FIELD_DATA_DEF
   unsigned char  cEyeCatcher[4];      // eye catcher
   short          sStructVer;          // version of this structure
   short          sStructLen;          // length of this structure
   char           bSrcNull;            // Source Null indicator
   char           bDstNull;            // Dest Null indicator
   int            iSrcBufLen;          // length, or max length of source buffer
                                       // in bytes. On Return in case bCopyToDest of SvcDef
                                       // is FALSE this variable will be updated to
                                       // contain the size of the output data in bytes
   int            iDstBufLen;          // length, or max length of dest buffer
                                       // in bytes.  On Return in case bCopyToDest of SvcDef
                                       // is TRUE this variable will be updated to
                                       // contain the size of the output data in bytes
   int            iOutBufLen;          // bytes of the data returned in the pDstBuf. In case of
                                       // ODPPSUCCESS, this is the bytes of return data. If
                                       // RETURN CODE = ODPP_DATACNV_ERR_DATA_TRUNCATED, this
                                       // is the max bytes required to hold the return data
   int            iErrorCode;          // Error code for this data
   unsigned char *pSrcBuf;             // buffer to source buffer
   unsigned char *pDstBuf;             // buffer to destination buffer
} DP_FIELD_DATA_DEF;
```

17.2.3 DP_ROW_DEF

```
//-------------------------------------------------------------------
// Name:          DP_ROW_DEF
// Description:
// The Row Definition encapsulates the DP_FIELD_DEFINE struct
// to provide a 2-dimensional array of DP_FIELD_DEFINE
//
//********************IMPORTANT*****************************************
// bProcessed should be FALSE (i.e. 0) for each row
// in the first Service Call.
//********************IMPORTANT*****************************************
//-------------------------------------------------------------------
typedef struct DP_ROW_DEF
{
   struct DP_ROW_DEF
                *pNext;               // Next element in chain of DP_ROW_DEF
   unsigned char  cEyeCatcher[4];     // eye catcher
   short          sStructVer;         // version of this structure
   short          sStructLen;         // length of this structure
   char           bProcessed;         // TRUE if the ROW processed earlier
                                      // This should be FALSE (i.e. 0) for each
                                      // row in first Service Call.
   char           bHasError;          // TRUE if any column(s) has an error code
   short          sCount;             // Count of DP_FIELD_DATA_DEF(s) either
                                      // in form of an array or a chain of
                                      // elements
   DP_FIELD_DATA_DEF
                *pFldDataDefine;      // Pointer to an array of
                                      // DP_FIELD_DATA_DEF(s) OR pointer
                                      // to the first element of chain of
                                      // DP_FIELD_DATA_DEF.
}DP_ROW_DEF;
```

17.2.4 DP_ROWSET_DEF

```
//-------------------------------------------------------------------
// Name:          DP_ROWSET_DEF
// Description:
// The Rowset Defintion encapsulates a batch of DP_ROW_DEFs
//-------------------------------------------------------------------
typedef struct DP_ROWSET_DEF
{
   unsigned char  cEyeCatcher[4];     // Eye catcher
   short          sStructVer;         // version of this structure
   short          sStructLen;         // length of this structure
   int            iCount;             // Count of DP_ROW_DEF(s) either in form of
                                      // an array or a chain of elements
   int            iLastPrcsRow;       // Index of the last row processed. Needed for
                                      // restart after discard limit failure.
                                      // This should be 0 for first Service Call.
   DP_ROW_DEF     *pRowDefine;        // Pointer to an array of DP_ROW_DEF(s)
                                      // OR pointer to the first element of chain of
                                      // DP_ROW_DEF.
}DP_ROWSET_DEF;
```

17.2.5 DPFD_WC_SS

```
//------------------------------------------------------------------------
// Name:          DPFD_WC_SS
// Description:
// The field definition for the wide-character (Unicode)
// sub-structure for DP_FIELD_DEF
//------------------------------------------------------------------------
typedef struct DPFD_WC_SS
{
   ODPP_WCHAR    *pColName;            // Pointer to column name
                                      // expressed in wide character (Unicode) format
}DPFD_WC_SS;
```

17.2.6 DPFD_MC_SS

```
//------------------------------------------------------------------------
// Name:          DPFD_MC_SS
// Description:
// The field definition for the mixed-character (e.g. SBCS/MBCS)
// sub-structure for DP_FIELD_DEF
//------------------------------------------------------------------------
typedef struct DPFD_MC_SS
{
   char           cColNameDBMSType;    // DBMS type of the column name
   int            iColNameCodePage;    // Code page of the column name
   //Set iColNameCodePage to ODPP_CODE_PAGE_NONE to use code page and DBMSType
   //supplied by the parent structure DP_SVC_DEF

   char          *pColName;            // Pointer to column name
                                      // expressed in mixed (SBCS/MBCS) character format
}DPFD_MC_SS;
```

17.2.7 DP_FIELD_DEF

```
//------------------------------------------------------------------------
// Name:          DP_FIELD_DEF
// Description:
// The field definition describes all of the attributes of a field
//------------------------------------------------------------------------
typedef struct DP_FIELD_DEF
{
   struct DP_FIELD_DEF
                 *pNext;              // Pointer to next element in chain of DP_FIELD_DEF
   char           cEyeCatcher[4];     // Eye catcher
   short          sStructVer;         // Version of this structure
   short          sStructLen;         // Length of this structure
   short          sDatatype;          // Data type of the input data
   int            iLength;            // Column length (for string types)
   int            iPrecision;         // Precision (for decimal types) and ORA_VARNUM
   short          sScale;             // Scale (for decimal data)
   int            iDataCodePage;      // Code page of the data
   char           cDataDBMSType;      // DBMS type of the data
   int            iColNameBytes;      // Size, in bytes, of the column name
   char           cSubType;           // Type of sub-structure
                                      // (W or blank or 0=wide-char, M=mixed (SBCS/MBCS) char
   union COLNAME_SS                   // Union of sub-structures
   {
      DPFD_WC_SS *pWC;                // Pointer to wide character (Unicode) sub-structure
      DPFD_MC_SS *pMC;                // Pointer to mixed character (SBCS/MBCS) sub-structure
   }CN;

} DP_FIELD_DEF;
```

17.2.8 DPPRM_NAME_WC_SS

```
//------------------------------------------------------------------------
// Name:          DPPRM_NAME_WC_SS
// Description:
// Wide-character (Unicode) parameter name sub-structure for DP_INIT_OP_DEF
//------------------------------------------------------------------------
typedef struct DPPRM_NAME_WC_SS
{
   ODPP_WCHAR    *pParmName;           // Pointer to parm key
                                       // expressed in wide character (Unicode) format
}DPPRM_NAME_WC_SS;
```

17.2.9 DPPRM_NAME_MC_SS

```
//------------------------------------------------------------------------
// Name:          DPPRM_NAME_MC_SS
// Description:
// mixed-character (e.g. SBCS/MBCS) parameter name sub-structure for
// DP_INIT_OP_DEF
//------------------------------------------------------------------------
typedef struct DPPRM_NAME_MC_SS
{
   char           cParamNameDBMSType;  // DBMS type of the parameter name
   int            iParamNameCodePage;  // Code page of the parameter name
   //Set iParamNameCodePage to ODPP_CODE_PAGE_NONE to use code page and DBMSType
   //supplied by the parent structure (DP_FRMW_PARAMS_DEF/DP_SVC_DEF)

   char          *pParmName;           // Pointer to parm key
                                       // expressed in mixed (SBCS/MBCS) character format
}DPPRM_NAME_MC_SS;
```

17.2.10 DPPRM_VAL_WC_SS

```
//------------------------------------------------------------------------
// Name:          DPPRM_VAL_WC_SS
// Description:
// Wide-character (Unicode) parameter value sub-structure for DP_INIT_OP_DEF
//------------------------------------------------------------------------
typedef struct DPPRM_VAL_WC_SS
{
   ODPP_WCHAR    *pParamVal;           // Pointer to parm key
                                       // expressed in wide character (Unicode) format
}DPPRM_VAL_WC_SS;
```

17.2.11 DPPR_VAL_MC_SS

```
//------------------------------------------------------------------------
// Name:          DPPRM_VAL_MC_SS
// Description:
// mixed-character (e.g. SBCS/MBCS) parameter value sub-structure for
// DP_INIT_OP_DEF
//------------------------------------------------------------------------
typedef struct DPPRM_VAL_MC_SS
{
   char           cParamValDBMSType;   // DBMS type of the parameter value
   int            iParamValCodePage;   // Code page of the parameter value
   //Set iParamValCodePage to ODPP_CODE_PAGE_NONE to use code page and DBMSType
   //supplied by the parent structure (DP_FRMW_PARAMS_DEF/DP_SVC_DEF)

   char          *pParamVal;           // Pointer to parm key
                                       // expressed in mixed (SBCS/MBCS) character format
}DPPRM_VAL_MC_SS;
```

17.2.12 DP_INIT_OP_DEF

```
//------------------------------------------------------------------------
// Name:          DP_INIT_OP_DEF
// Description:
// Init Parameter Definition contains all the initialization
// parameters required for the current service
//------------------------------------------------------------------------
typedef struct DP_INIT_OP_DEF
{
   struct DP_INIT_OP_DEF
                *pNext;               // Pointer to next element in chain of parameters
   char          cEyeCatcher[4];      // Eye Catcher
   short         sStructVer;          // Version of this structure
   short         sStructLen;          // Length of this structure
   unsigned short usParameterID;      // ODPP Parameter

   char          cParamNameSubType;   // Type of parameter sub-structure
                                      // (W or blank=wide-char, M=mixed (SBCS/MBCS) char

   int           iParamNameBufBytes;  // Size (in bytes) of param value buffer

   // =================
   // Parameter name
   // =================
   // Used in scenario when there is no parameter ID
   // Set usParameterID = 0 when a parameter name is passed instead of ParameterID
   union PARAM_NAME_SS
   {
      DPPRM_NAME_WC_SS
                *pWC;                 // Pointer to wide character (Unicode) sub-structure
      DPPRM_NAME_WC_SS
                *pMC;                 // Pointer to mixed character (SBCS/MBCS) sub-structure
   }PN;

   // =================
   // Parameter value
   // =================
   int           iValueSubType;       // Type of parameter value as defined below
                                      // any value except these are user defined values
                                      // and indicates the use of pUserVal.
                                      //=============================================
                                      // Custom service providers can define their own
                                      // values for their usage
                                      //=============================================
                #define PARAM_VAL_NONE  0  //This paramaeter has no value
                #define PARAM_VAL_NUM   1  //Parameter value is numeric
                #define PARAM_VAL_WC    2  //Parameter value is Wide Char string
                #define PARAM_VAL_MC    3  //Parameter value is SBCS/MBCS

   int           iValueBufBytes;      // Size (in bytes) of param value buffer

   union PARAM_VAL
   {
      unsigned int
                uiVal;                // User-supplied numeric value
      DPPRM_VAL_WC_SS
                *pWC;                 // Pointer to wide character (Unicode) sub-structure
      DPPRM_VAL_MC_SS
                *pMC;                 // Pointer to mixed character (SBCS/MBCS) sub-structure
      void      *pUserVal;            // User defined parameter value
   }PV;

}DP_INIT_OP_DEF;
```

17.2.13 DP_FRMW_PARAMS_DEF

```
//------------------------------------------------------------------------
// Name:          DP_FRMW_PARAMS_DEF
// Description:
// Encapsulates the parameters required for the
// ODPP framework initialization
//------------------------------------------------------------------------
typedef struct DP_FRMW_PARAMS_DEF
{
   char           cEyeCatcher[4];      // Eye Catcher
   short          sStructVer;          // version of this structure
   short          sStructLen;          // Length of this structure
   short          sParamCount;         // Count of parameters in pParams

   // Default Code Page and DBMS Type for Parameter names and values
   //
   // Note: Codepage/DBMSType values set for individual parameters take
   // precedence over these value
   char           cParamValDBMSType;   // DBMSType of parameter value
   int            iParamValCP;         // Code page of parameter value

   DP_INIT_OP_DEF
                 *pParams;             // List of parameters

}DP_FRMW_PARAMS_DEF;
```

17.2.14 DP_SVC_DEF

```
//------------------------------------------------------------------------
// Name:          DP_SVC_DEF
// Description:
// The Service Definition contains the parameters and field definition lists
// and the other service definition type initialization parameters.
//------------------------------------------------------------------------
typedef struct DP_SVC_DEF
{
   char           cEyeCatcher[4];      // Eye Catcher
   short          sStructVer;          // version of this structure
   short          sStructLen;          // length of this structure
   char           bRefreshSVCDef;      // [For Future Use] When TRUE, this
                                       // causes the existing the existing
                                       // service definition to be updated
   char           bCopyToDest;         // Copy the masked data to destination buf
   short          sParamCount;         // Count of elements in pParams
   short          sFldCount;           // Count of elements in pFldDef

   // Default Code Page and DBMS Type for Parameter names and values
   //
   // Note: Codepage/DBMSType values set for individual parameters/fields
   // take precedence over these value
   char           cCtrlDataDBMSType;   // DBMSType of control Data
                                       // in DP_INIT_OP_DEF and DP_FIELD_DEF
   int            iCtrlDataCP;         // Code page of control data
                                       // in DP_INIT_OP_DEF and DP_FIELD_DEF

   DP_INIT_OP_DEF
                 *pParams;             // Pointer to array or first element in
                                       // chain of DP_INIT_OP_DEF. Provides all
                                       // needed parameters.
   DP_FIELD_DEF  *pFldDef;             // Pointer to array or first element in
                                       // chain of DP_FIELD_DEF.

}DP_SVC_DEF;
```

17.2.15 DPPD_WC_SS

```
//------------------------------------------------------------------------
// Name:          DPPD_WC_SS
// Description:
// The provider definition for the wide-character (Unicode)
// sub-structure for DP_PRV_DETAILS
//------------------------------------------------------------------------
typedef struct DPPD_WC_SS
{
   ODPP_WCHAR    *pProviderName;       // Pointer to provider name
                                       // expressed in wide character (Unicode) format
   ODPP_WCHAR    *pDescription;        // Pointer to provider description
                                       // expressed in wide character (Unicode) format
}DPPD_WC_SS;
```

17.2.16 DPPD_MC_SS

```
//------------------------------------------------------------------------
// Name:          DPPD_MC_SS
// Description:
// The provider definition for the mixed-character (e.g. SBCS/MBCS)
// sub-structure for DP_PRV_DETAILS
//------------------------------------------------------------------------
typedef struct DPPD_MC_SS
{
   char          *pProviderName;       // Pointer to provider name
                                       // expressed in mixed (SBCS/MBCS) character format
   char          *pDescription;        // Pointer to provider description
                                       // expressed in mixed (SBCS/MBCS) character format
}DPPD_MC_SS;
```

17.2.17 DP_PRV_DETAILS

```
//------------------------------------------------------------------------
// Name:          DP_PRV_DETAILS
// Description:
// The Provider Details is an entry in the linked-list of
// provider-type information that is returned
// to the caller of Provider_Enumerate
//------------------------------------------------------------------------
typedef struct DP_PRV_DETAILS
{
   struct DP_PRV_DETAILS
                 *pNext;               // Pointer to next element in chain
   char           bUserDefine;         // When TRUE, this is a user-supplied provider
   char           bLicensed;           // When TRUE, this is a licensed provider
   short          sProviderID;         // ODPP internally defined provider ID
   short          sProviderVer;        // Version of the provider
   int            iPrvNameBufBytes;    // Size, in bytes, of the provider name
   int            iPrvDescBufBytes;    // Size, in bytes, of the provider description
   union PD_SS
   {
      DPPD_WC_SS *pWC;                 // Pointer to wide character (Unicode) sub-structure
      DPPD_MC_SS *pMC;                 // Pointer to mixed character (SBCS/MBCS) sub-structure
   }PD;
}DP_PRV_DETAILS;
```

17.2.18 DP_PRV_DEF

```
//------------------------------------------------------------------------
// Name:          DP_PRV_DEF
// Description:
// The Provider Definition contians provider details in a linked-list of
// provider-type information that is returned to the caller of
// Provider_Enumerate
//------------------------------------------------------------------------
typedef struct DP_PRV_DEF
{
   char           cEyeCatcher[4];      // Eye Catcher
   short          sStructVer;          // Version of this structure
   short          sStructLen;          // Length of this structure
   char           cSubType;            // Type of sub-structure
                                       // (W or blank=wide-char, M=mixed (SBCS/MBCS) char
   char           cPDDBMSType;         // DBMSType of the provider name and description
   int            iPDCodePage;         // Code page of the provider name and description

   DP_PRV_DETAILS
                 *pPrvDetails;         // Array or LinkList of Providers with details
}DP_PRV_DEF;
```

17.2.19 DPEM_WC_SS

```
//------------------------------------------------------------------------
// Name:          DPEM_WC_SS
// Description:
// The Formatted error message definition for the wide-character (Unicode)
// sub-structure for DP_ERR_MSG
//------------------------------------------------------------------------
typedef struct DPEM_WC_SS
{
   int            iMsgSrcBytes;        // Size, in bytes, of the error message source
   int            iMsgBytes;           // Size, in bytes, of the error message description
   int            iMsgBodyBytes;       // Size, in bytes, of the error message body
   ODPP_WCHAR    *pMsgSrc;             // Pointer to error message source
                                       // expressed in wide character (Unicode) format
   ODPP_WCHAR    *pMsg;                // Pointer to error message description
                                       // expressed in wide character (Unicode) format
   ODPP_WCHAR    *pMsgBody;            // Pointer to error message body
                                       // expressed in wide character (Unicode) format
}DPEM_WC_SS;
```

17.2.20 DPEM_MC_SS

```
//------------------------------------------------------------------------
// Name:          DPEM_MC_SS
// Description:
// The provider definition for the mixed-character (e.g. SBCS/MBCS)
// sub-structure for DP_ERR_MSG
//------------------------------------------------------------------------
typedef struct DPEM_MC_SS
{
   char           cEMDBMSType;         // DBMS type of the error message details
   int            iEMCodePage;         // Code Page of the error message details
   int            iMsgSrcBytes;        // Size, in bytes, of the error message source
   int            iMsgBytes;           // Size, in bytes, of the error message description
   int            iMsgBodyBytes;       // Size, in bytes, of the error message body
   char          *pMsgSrc;             // Pointer to error message source
                                       // expressed in mixed (SBCS/MBCS) character format
   char          *pMsg;                // Pointer to error message description
                                       // expressed in mixed (SBCS/MBCS) character format
   char          *pMsgBody;            // Pointer to error message body
                                       // expressed in mixed (SBCS/MBCS) character format
}DPEM_MC_SS;
```

17.2.21 DP_ERR_MSG

```
//------------------------------------------------------------------------
// Name:          DP_ERR_MSG
// Description:
// The Provider Defintioner is an entry in the linked-list of
// provider-type information that is returned
// to the caller of Provider_GetFormattedErrorMsg
//------------------------------------------------------------------------
typedef struct DP_ERR_MSG
{
   char           cEyeCatcher[4];      // Eye Catcher
   short          sStructVer;          // Version of this structure
   short          sStructLen;          // Length of this structure
   int            iErrNum;             // Error number/Return code. USED ONLY
                                       // WHEN pECB is NULL, IGNORED OTHERWISE
   int            iLanguage;           // Language to use for error messages
   DP_ECB        *pECB;                // Error Control Block
   char           cSubType;            // Type of sub-structure
                                       // (W or blank=wide-char, M=mixed (SBCS/MBCS) char
   union EM_SS
   {
      DPEM_WC_SS                       // Pointer to wide character (Unicode) sub-structure
                 *pWC;
      DPEM_MC_SS                       // Pointer to mixed character (SBCS/MBCS) sub-structure
                 *pMC;
   } EM;
}DP_ERR_MSG;
```