

Database Access to Call Detail Records

MX Enterprise Media Exchange

1 Introduction

A call detail record (CDR) on the MX250 or the MX1200 is information about a call made on the system. This information resides on the system's internal database, which can be accessible by any external application that performs standard ODBC access. The MX Administration software accesses this database to generate predefined reports from specific combinations of CDRs. Alternatively, administrators can generate custom reports or create triggers for other applications by directly accessing the CDR database.

This document describes the architecture of CDRs for developers who need direct access to the internal CDR database on the MX250 and MX1200 systems. MySQL 4.0 is used as the database server.

2 Database Schema

The CDR database schema is shown in figure 1. The 12 accessible tables for CDRs are:

aainput. contains information about activity of user inputs accessed from the automated attendants

acdgroup. contains all advanced ACD groups

agentlogin. contains information about ACD agent login and logout activity

billablecall. contains information about all connections to external parties

callbackattempt. contains information about call back attempts

callbackrequest. contains all of the call back requests

customfieldheader. contains descriptions of the **mxuser.CustomX** fields

extension. contains all extensions for temporal users

mxuser. table with temporal user information

responsibleparty. contains information about parties, responsible for the billable calls

session. the main table in CDR database, all sessions which were made are stored here

userpresence. contains information about user and agents presence states

3 Definitions

3.1 Sessions

Every call made or received on the MX system (internal and external) is associated with a *session*. Basically, a session is a collection of detailed information about the call. Technically, a session comprises content of database fields store information like the date and time, duration, type of call, calling party details, and called party details.

Details about all sessions on a system are contained in the **Sessions** table, which holds up to 100,000 records. When this table is full, new records overwrite the oldest records.

Database Access to Call Detail Records

MX Enterprise Media Exchange

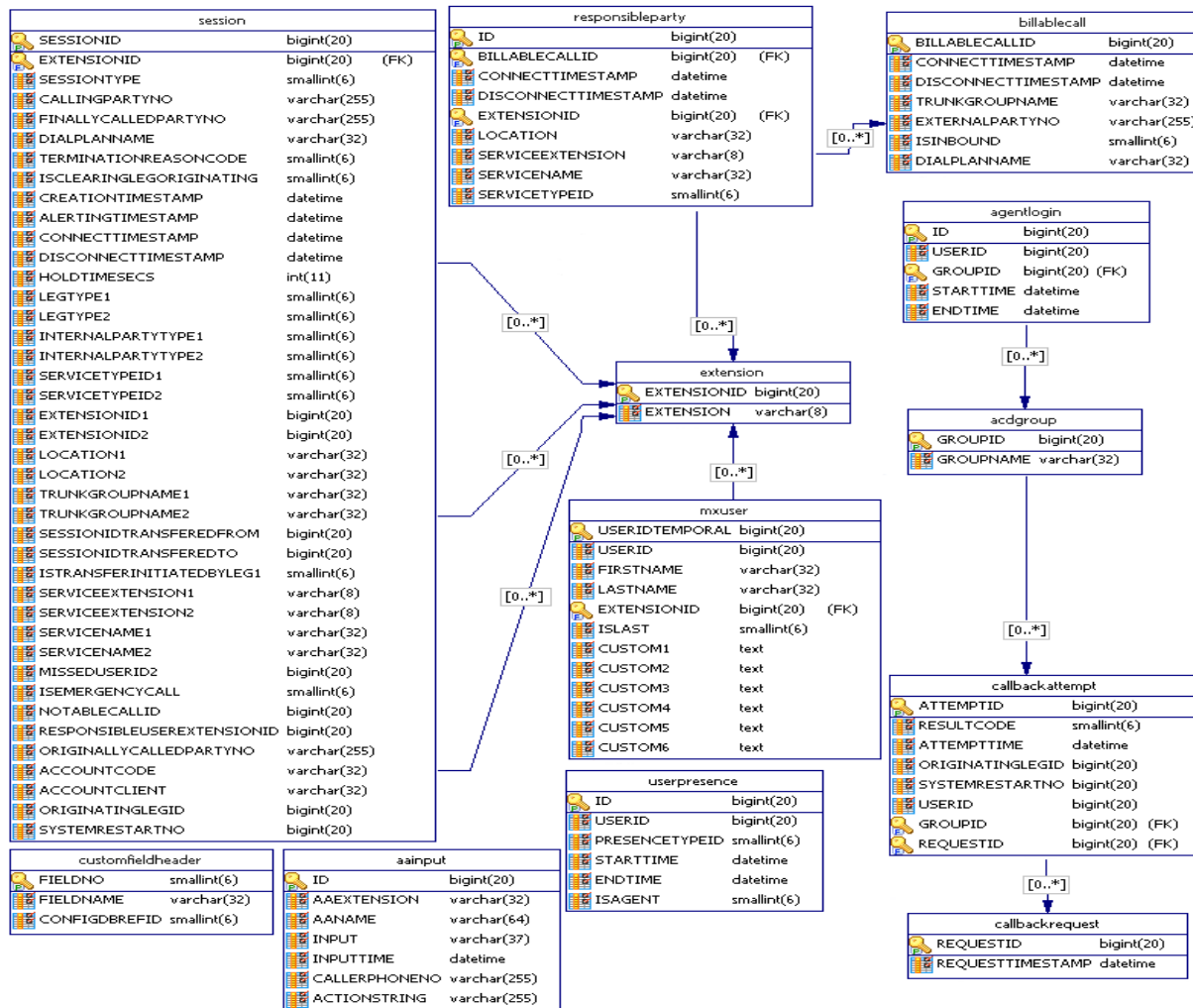


Figure 1 Database Schema

3.2 Legs

A session requires two *legs*, the *originating leg* and the *terminating leg*. This is a similar concept to having two sides of a call, the calling side and the called side. Basically, a leg is a collection of detailed information about the calling or called side. Technically, a leg comprises content of database fields that store information like type of leg (internal or external source), phone number, trunk line used, and device used.

Database Access to Call Detail Records

MX Enterprise Media Exchange

Within the **sessions** table, field names ending with the number **1** or **2** indicate that the associated field contains information about a leg. Field names ending with **1** represent content for the originating leg (calling side). Field names ending with **2** represent content for the terminating leg (called side).

3.3 Temporal User Information

The **mxuser** and **extension** tables contain fields that describe users who create sessions on the MX system. Every session is associated with a single record in **extension** and a single record in **mxuser**. These tables store both current and previous information about the user, which provide for tracking of the user information at the time the session was created. For every record in **extension**, there is referenced one and only one record in **mxuser**.

The first session with which a user is involved will create a record in the **mxuser** and a record in **extension**. The two linked records contain information about the user's login ID, name, and extension. All subsequent sessions involving the same user will be associated with the same pair of linked records until the system administrator changes the user's ID, name, or extension. The first session created involving the user after a change in his or her user information will create a new entry in the **mxuser** table, and all subsequent sessions involving the user will be associated with the new record until the next change in user information.

3.4 Billable Call and Responsible Party

A *billable call* is a call that has an external source for one of the legs. The duration of a single billable call is the difference between the end time of the last session (including all transfers and merges of the external leg), and the start time of the first session that involves the external leg. Details about billable calls are mostly stored in the **billablecall** table.

The *responsible party* of a billable call is the internal user or service that make up the other leg of the session. Details about the responsible party is stored in the **responsibleparty** table.

4 Table Descriptions

4.1 aainput

The **aainput** table contains information about a caller's input during a session with an automated attendant. This table is populated only if the checkbox *Record user input to CDR* is enabled for a dialogue within the automated attendant script, as shown in figure 2. The description for fields in **aainput** is shown in figure 3.

Database Access to Call Detail Records

MX Enterprise Media Exchange

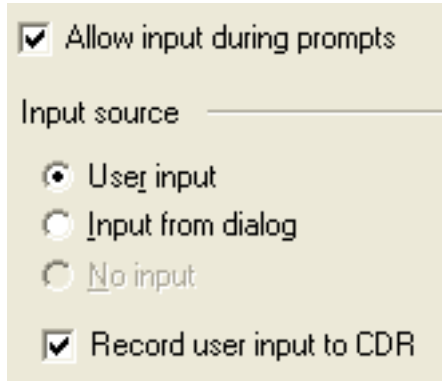


Figure 2 Enabling aainput Table on Auto Attendant Dialogue

Field Name	Purpose	Data Type	Index	Values	Relations
ID	Primary key	BigInt(20)	None	Automatically generated	None
AAExtension	Extension of automated attendant	VarChar(32)	BYNAMEANDINPUT TIME	Restricted by data type	None
AAName	Name of automated attendant	VarChar(64)	BYNAMEANDINPUT TIME	Restricted by data type	None
Input	User input recorded from automated attendant	VarChar(37)	None	Restricted by data type	None
InputTime	Date and time of record	DateTime	BYNAMEANDINPUT TIME	Restricted by data type	None
CallerPhoneNo	Caller ID for the caller that accessed the user input that was recorded	VarChar(255)	None	Restricted by data type	None
ActionString	Action type and configured parameters	VarChar(255)	None	Restricted by data type	None

Figure 3 aainput Table

4.2 acdgroup

The **acdgroup** table contains the name of the ACD group to which a user belongs at the time the user is involved in a session pertaining to that group. The description for fields in **acdgroup** is shown in figure 4.

Field Name	Purpose	Data Type	Index	Values	Relations
GroupID	Primary key	BigInt(20)	None	Automatically generated	None
GroupName	Name of the ACD group	VarChar(32)	None	Restricted by data type	None

Figure 4 acdgroup Table

Database Access to Call Detail Records

MX Enterprise Media Exchange

4.3 agentlogin

The **agentlogin** table contains the dates and times when agents log into and out of their ACD group. The description for fields in **agentlogin** is shown in figure 5.

Field Name	Purpose	Data Type	Index	Values	Relations
ID	Primary key	BigInt(20)	None	Automatically generated	None
UserID	Reference to mxuser.UserID	BigInt(20)	ALBYUSERID	Restricted by data type	Many to one with mxuser.UserID
GroupID	Reference to acdgroup.GroupID	BigInt(20)	ALBYGROUPID	Restricted by data type	Many to one with acdgroup.GroupID
StartTime	ACD agent login time	DateTime	ALBYSTARTIME	Restricted by data type	None
EndTime	ACD agent log off time	DateTime	None	Restricted by data type	None

Figure 5 agentlogin Table

4.4 billablecall

The **billablecall** table contains information about billable calls, including start time, end time, trunk group, and external party number. If a call involves two external legs as the result of a transfer or call handling rules, two billable calls are created. In this case the responsible party for both billable calls will be the user who initiated the transfer, merge, or active call handling rule. Information about the internal parties involved with the sessions of a billable call is stored in the **responsibleparty** table. Because a billable call comprises sessions, some information in **billablecall** table is duplicated in the **session** table. The description for fields in **billablecall** is shown in figure 6.

Field Name	Purpose	Data Type	Index	Values	Relations
BillableCallId	Primary Key	BigInt(20)	None	Automatic ally generated	None
ConnectTimestamp	Contains date and time when connection to the external party began	DateTime	BYTIMEANDINBOUND ONCONNECT	Restricted by data type	None
DisconnectTimestamp	Contains date and time when connection to the external party ended	DateTime	BYTIMEANDINBOUND	Restricted by data type	None
TrunkGroupName	Contains name of trunk group	VarChar(32)	BYTRUNKGROUPNAME	Restricted by data type	None
ExternalPartyNo	Contains called phone number or address of external party	VarChar(255)	None	Restricted by data type	None

Figure 6 billablecall Table

Database Access to Call Detail Records

MX Enterprise Media Exchange

Field Name	Purpose	Data Type	Index	Values	Relations
IsInbound	Determines if current billable call is inbound	SmallInt(6)	BYTIMEANDINBOUND	0 – False 1 – True	None
DialPlanName	Contains dial plan route name	VarChar(32)	None	Restricted by data type	None

Figure 6 billablecall Table (Continued)

4.5 callbackattempt

The **callbackattempt** table contains information about call back attempts. A record in this table may reference a record in the **session** table if a session has been created from the **callback** window in the agent’s MXIE (shown in figure 7). The description for fields in **callbackattempt** is shown in figure 8.

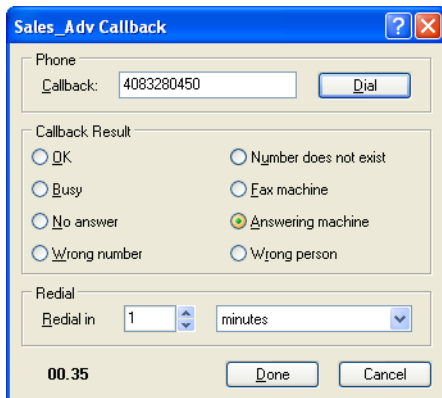


Figure 7 Callback Window in Agent’s MXIE

Field Name	Purpose	Data Type	Index	Values	Relations
ID	Primary key	BigInt(20)	None	Automatically generated	None
ResultCode	The result of the call back attempt	SmallInt(6)	None	0 – Unknown 1 – OK 2 – Busy 3 – No answer 4 – Wrong number 5 – Number not exists 6 – Fax machine 7 – Answer Machine 8 – Wrong person 9 – Cancelled 10 – Time-out	None
AttemptTime	Time and date the call was made	DateTime	None	Restricted by data type	None

Figure 8 callbackattempt Table

Database Access to Call Detail Records

MX Enterprise Media Exchange

Field Name	Purpose	Data Type	Index	Values	Relations
OriginalLedID	Reference to session table	BigInt(20)	None	Restricted by reference table	Reference to session.OriginallyLedID
SystemRestartNo	Reference to session table	BigInt(20)	None	Restricted by reference table	Reference to session.SystemRestartNo
UserID	Reference to session table	BigInt(20)	None	Restricted by reference table	Many to one with mxuser.UserID
GroupID	Reference to mxuser table	BigInt(20)	None	Restricted by reference table	Many to one with acdgroup.GroupID

Figure 8 callbackattempt Table (Continued)

4.6 callbackrequest

The **callbackrequest** table contains records of all call back requests. The field description of **callbackrequest** is shown in figure 9.

Field Name	Purpose	Data Type	Index	Values	Relations
RequestID	Primary key	BigInt(20)	UPBYSTARTTIME	Automatically generated	None
RequestTimeStamp	Time and date that the call back request was initiated	DateTime	None	Restricted by data type	None

Figure 9 callbackrequest Table

4.7 customfieldheader

The **customfieldheader** table contains descriptions of the **mxuser.CustomX** fields. For example *Custom1* field in **mxusers** has the name and description taken from **customfieldheader** with *FieldNo* having the value 1. The field description of **customheaderfield** is shown in figure 10.

Field Name	Purpose	Data Type	Index	Values	Relations
FieldNo	Primary key	BigInt(20)	UPBYSTARTTIME	Automatically generated	None
FieldName	Name of mxusers custom field	VarChar(32)	None	Restricted by data type	None
ConfigDBRefID	Internal use	SmallInt(6)	None	Restricted by data type	None

Figure 10 customheaderfield Table

Database Access to Call Detail Records

MX Enterprise Media Exchange

4.8 extension

The **extension** table contains temporal extension numbers of internal users or services. For every record in **extension**, there is a corresponding record in **mxuser**. The description for fields in **extension** is shown in figure 11.

Field Name	Purpose	Data Type	Index	Values	Relations
ExtensionId	Primary key	BigInt (20)	UPBYSTARTTIME	Automatically generated	None
Extension	Contains extension name	VarChar(8)	BYEXTENSIONTEXT	0 – Unknown 1 – Voice 2 – Chat	None

Figure 11 extension Table

4.9 mxuser

The **mxuser** table contains information about temporal users associated with sessions. For every record in **mxuser**, there is a corresponding record in **extension**. The field description of **mxuser** is shown in figure 12.

Field Name	Purpose	Data Type	Index	Values	Relations
UserIdTemporal	Primary Key	BigInt(20)	UPBYSTARTTIME	Automatically generated	None
UserId	UserID	BigInt(20)	BYUSERID	Restricted by data type	None
FirstName	User's first name	VarChar(32)	None	Restricted by data type	None
LastName	User's last name	VarChar(32)	None	Restricted by data type	None
ExtensionId	Reference to extension table	BigInt(20)	BYEXTENSIONID	Restricted by data type	Relation many-to-one on extension.ExtensionId
IsLast	If this field is true, then the corresponding ExtensionID is the latest and should be used for referencing other tables where ExtensionID is required.	SmallInt(6)	None	0 – False 1 – True	None
CustomX	To mark current record of the user	Text	None	Restricted by data type	See definition of customfieldheader table

Figure 12 mxuser Table

Database Access to Call Detail Records

MX Enterprise Media Exchange

4.10 responsibleparty

The **responsibleparty** table contains information about internal parties, which are responsible for the billable calls (**billablecall** table). The description for fields in **responsibleparty** is shown in figure 13.

Field Name	Purpose	Data Type	Index	Values	Relations
Id	Primary key	BigInt(20)	UPBYSTARTTIME	Automatically generated	None
BillableCallId	Reference to billable call (billablecall table)	BigInt(20)	INDEX_FK	Restricted by data type	Relations many-to-one on billablecall.BillableCallId
ConnectTimestamp	Contains date and time when billable call is started	DateTime	BYTIME	Restricted by data type. Null or 0 if the call is not answered.	None
DisconnectTimestamp	Contains date and time when billable call is ended	DateTime	BYTIME	Restricted by data type	None
ExtensionId	This field contains reference to extension table	BigInt(20)	BYEXTID	Restricted by data type	Relation many-to-one on extension.ExtensionId
Location	Responsible party location	VarChar(32)	ONLOCATION	Restricted by data type	None
ServiceExtension	This field contains party's extension	VarChar(8)	None	Restricted by data type	None
ServiceName	This field contains service's name	VarChar(32)	None	Restricted by data type	None
ServiceTypeId	This field contains type of service	SmallInt(6)	None	0 – Operator 1 – AA 2 – ACD	None

Figure 13 responsibleparty Table

4.11 session

The **session** table contains details about a session. Each call on the system creates at least one record on this table. Calls can be classified by type and by direction. Classifying calls by type allow you to determine whether or not a call involves an external leg. Classifying calls by direction allow you to determine whether or not a call was made or received into the system.

There are two types possible for a call:

1. **Internal:** Both legs are internal.
2. **External:** At least one leg is external.

Database Access to Call Detail Records

MX Enterprise Media Exchange

There are two directions possible for a call:

1. **Inbound:** For a given extension *PartyExt*, the call is inbound if this party (user or service) received the call (present in leg 2).
 - User, if the *InternalPartyType2* value is 0 and *extension.ExtensionId* value is equal to *session.ExtensionId2* and *extension.Extension* value is *PartyExt*.
 - Service, if the *InternalPartyType2* value is 1 and *ServiceExtension2* value is equal to *PartyExt*.
2. **Outbound:** For a given extension *PartyExt* the call is Outbound if this party (user or service) initiated the call (present in leg1).
 - User, if *InternalPartyType1* value is 0 and *extension.ExtensionId* value equal to *session.ExtensionId1* and *extension.Extension* value is *PartyExt*.
 - Service, if *InternalPartyType1* value is 1 and *ServiceExtension1* value is *PartyExt*

Every user and or party has an extension. If leg X belongs to a user, the field *session.ExtensionIdX* is the reference to the **extension** table, and the **mxuser** table (from which you can get all information about user) has a reference to the **extension** table.

If a session has two external legs (result of a transfer or call handling rules) it is split between two billable calls (since two external connections are present). In this case the responsible party for both of these billable calls will be the user who initiated the transfer/merge or the author of call handling rules. All original and split-sessions are stored in **session** table.

Unanswered calls will have a ConnectTimeStamp of "0" time, which is saved using GMT as "1970/01/01" using year/month/day format. If you want to exclude unanswered calls from your reports, your SQL statements need to exclude the above ConnectTimeStamp.

The field description for session is shown in figure 14.

Field Name	Purpose	Data Type	Index	Values	Relations
SessionId	Primary key	BigInt(20)	SESSION_PK	Automatically generated	None
SessionType	Describes the type of the session	SmallInt(6)	None	0 – Unknown 1 – Voice 2 – Chat	None
CallingPartyNo	Contains dialer's phone number or address	VarChar(255)	None	Restricted by data type	None
FinnalyCalledPartyNo	Contains called phone number or address	VarChar(255)	None	Restricted by data type	None
DialPlanName	Contains dial plan route name	VarChar(32)	None	Restricted by data type	None
TerminationReasonCode	Reserved	N/A	N/A	N/A	N/A
IsClearingLegOriginating	Indicates whether the originating leg hanged up	SmallInt(6)	None	0 – False 1 – True	None

Figure 14 session Table

Database Access to Call Detail Records

MX Enterprise Media Exchange

Field Name	Purpose	Data Type	Index	Values	Relations
CreationTimestamp	Reserved	N/A	N/A	N/A	N/A
AlertingTimeStamp	Reserved	N/A	N/A	N/A	N/A
ConnectTimestamp	Contains date and time when current session started	DateTime	ONDATE	Restricted by data type	None
DisconnectTimestamp	Contains date and time when current session ended	DateTime	None	Restricted by data type	None
HoldTimeSecs	Contains count of seconds session was hold	Int(11)	None	Restricted by data type	None
LegType1	Contains the originating leg	Smallint(6)	None	0 – Undefined 1 – Internal 2 – External 3 – VM client	None
LegType2	Contains the terminating leg	SmallInt(6)	None	0 – Undefined 1 – Internal 2 – External 3 – VM client	None
InternalPartyType1	Contains originating type of party. This field is only populated for internal calls.	SmallInt(6)	None	0 – User 1 – Service	None
InternalPartyType2	Contains originating type of party. This field is only populated for internal calls.	SmallInt(6)	None	0 – User 1 – Service	None
ServiceTypeId1	Contains originating type of leg's service. This field is only populated when InternalPartyTypeX = 1.	SmallInt(6)	None	0 – Operator 1 – VM 2 – AA 3 – ACD 4 – Oper. VM 5 – ACD VM 6 – Bind Server 7 – Park Server 8 – Page Server 9 – Hunt Group 10 – Adv. ACD	None

Figure 14 session Table (Continued)

Database Access to Call Detail Records

MX Enterprise Media Exchange

Field Name	Purpose	Data Type	Index	Values	Relations
ServiceTypeId2	Contains originating type of leg's service. This field is only populated when InternalPartyTypeX = 1.	SmallInt(6)	None	0 – Operator 1 – VM 2 – AA 3 – ACD 4 – Oper. VM 5 – ACD VM 6 – Bind Server 7 – Park Server 8 – Page Server 9 – Hunt Group 10 – Adv. ACD	None
ExtensionId1	ExtensionId1 is reference to extension.ExtensionId. This field is only populated if the call leg belongs to user..	BigInt(20)	BYEXTENSION1	Restricted by data type	Relationship many-to-one to extension.ExtensionId
ExtensionId2	ExtensionId1 is reference to extension.ExtensionId. This field is only populated if the call leg belongs to user..	BigInt(20)	BYEXTENSION2	Restricted by data type	Relationship many-to-one to extension.ExtensionId
Location1	Contains location of originating leg	VarChar(32)	None	Restricted by data type	None
Location2	Contains location of terminating leg	VarChar(32)	None	Restricted by data type	None
TrunkGroupName1	Contains name of trunk group or originating leg. This field is only populated for external calls.	VarChar(32)	None	Restricted by data type	None
TrunkGroupName2	Contains name of trunk group or originating leg. This field is only populated for external calls.	VarChar(32)	None	Restricted by data type	None
SessionIdTransferredFrom	If session is transferred, this field contains SessionId of transferring session.	BigInt(20)	None	Restricted by data type	Relation one-to-one on session.SessionId
SessionIdTransferredTo	If session is transferring, this field contains SessionId of transferred session.	BigInt(20)	None	Restricted by data type	Relation one-to-one on session.SessionId

Figure 14 session Table (Continued)

Database Access to Call Detail Records

MX Enterprise Media Exchange

Field Name	Purpose	Data Type	Index	Values	Relations
IsTransferInitiatedByLeg1	Determines if transfer is initiated by Leg 1	SmallInt(6)	None	0 – False 1 – True	None
ServiceExtension1	Contains extension of originating leg. These fields is only populated when Party X is service (InternalPartyTypeX = 1)	VarChar(8)	None	Restricted by data type	None
ServiceExtension2	Contains extension of terminating leg. These fields is only populated when Party X is service (InternalPartyTypeX = 1)	VarChar(8)	None	Restricted by data type	None
ServiceName1	Contains service name of originating leg. This field is only populated when Party X is service (InternalPartyTypeX = 1)	VarChar(32)	None	Restricted by data type	None
ServiceName2	Contains service name of terminating leg. This field is only populated when Party X is service (InternalPartyTypeX = 1)	VarChar(32)	None	Restricted by data type	None
MissedUserId	Reserved	N/A	N/A	N/A	N/A
IsEmergencyCall	If true then the call was made to an emergency dial plan entry.	SmallInt(6)	None	0 – False 1 – True	None
ResponsibleUserExtensionId	This field contains reference to extension table of responsible extension. This field is populated when a call is forwarded using call handling rule.	BigInt(20)	None	Restricted by data type	Relation many-to-one to extension.ExtensionId

Figure 14 session Table (Continued)

Database Access to Call Detail Records

MX Enterprise Media Exchange

Field Name	Purpose	Data Type	Index	Values	Relations
OriginallyCalledPartyNo	Contains the called party number for calls transferred by call handling rules	VarChar(255)	None	Restricted by data type	None
AccountCode	Contains the account code if used for the call	VarChar(32)	ACCOUNTCODES	Restricted by data type	
AccountClient	Name associated with the account code	VarChar(32)	ACCOUNTCODES	Restricted by data type	
OriginatingLegID	Reference to callbackattempt table	BigInt(20)	None	Restricted by data type	Relation one-to-many to callbackattempt. OriginatingLegID
SystemRestartNo	Reference to callbackattempt table	BigInt(20)	None	Restricted by data type	Relation one-to-many to callbackattempt. OriginatingLegID

Figure 14 session Table (Continued)

4.12 userpresence

The **userpresence** table contains presence information for users. The field description of **userpresence** is shown in figure 15.

Field Name	Purpose	Data Type	Index	Values	Relations
ID	Primary key	BigInt(20)	None	Automatically generated	None
UserID	Reference to mxuser.UserID	BigInt(20)	UPBYUSERID	Restricted by data type	Many to one with mxuser.UserID
PresenceTypeID	Presence type	SmallInt(6)	No	0 – Logged out 1 – Available 2 – Not available 3 – Busy 4 – At lunch 5 – In a meeting 6 – Be right back 7 – Appear offline 8 – Wrap up 9 – Active 10 – On the phone	Many to one with acdgroup.GroupID
StartTime	Time the presence state starts	DateTime	UPBYSTARTTIME	Restricted by data type	None
EndTime	Time the presence state ends	DateTime	None	Restricted by data type	None
IsAgent	Indicates if presence state is for an ACD agent	SmallInt(6)	None	0 – False 1 – True	None

Figure 15 userpresence Table

Database Access to Call Detail Records

MX Enterprise Media Exchange

5 Determining Call Statistics

5.1 Introduction

Several SQL statements are included to demonstrate how call detailed information can be extracted from the CDR database.

When creating SQL statements for MySQL it is important to note that table names are case sensitive. All of the CDR database tables are lowercase.

Table field names are not case sensitive.

5.2 Account Codes

5.2.1 References

Table(s): session

Field(s):

AccountCode
AccountClient

5.2.2 Sample SQL Statement

The following SQL statement will return all calls that were made using account codes and the associated account code name.

```
SELECT
    AccountCode, AccountClient
FROM
    session
WHERE
    AccountCode IS NOT NULL
ORDER BY
    AccountCode
```

5.3 Calls Abandoned by Caller while in ACD Queue

5.3.1 References

Table(s): session

Field(s):

ServiceTypeID2
ExtensionID2
SessionIDTransferredTo
IsClearingLegOriginating

Database Access to Call Detail Records

MX Enterprise Media Exchange

5.3.2 Sample SQL Statement

The following SQL statement will indicate the total number of calls that were abandoned by the caller while in the ACD queue.

```
SELECT COUNT(*)
FROM
    session
WHERE
    (ServiceTypeID2 = 3 OR ServiceTypeID = 10)
    AND ExtensionID2 IS NULL
    AND SessionIDTransferredTo = 0
    AND IsClearingLegOriginating = 1
```

5.4 Calls Made and Received by ACD Agents

5.4.1 References

Table(s): session

Field(s):

ExtensionID1
ServiceTypeID1
ServiceTypeID2

5.4.2 Sample SQL Statement

The following SQL statement will return the total number of call attempts, and calls made and received by ACD agents.

No distinction is made about the direction or the name of the ACD group.

```
SELECT
    COUNT(*)
FROM
    session
WHERE
    (ServiceTypeID1 = 3 AND ExtensionID1 IS NOT NULL)
    OR (ServiceTypeID2 = 3 AND ExtensionID2 IS NOT NULL)
```

5.5 Calls Made and Received by Operators

5.5.1 References

Table(s): session

Database Access to Call Detail Records

MX Enterprise Media Exchange

Field(s):

- ExtensionID1
- ServiceTypeID1
- ServiceTypeID2

5.5.2 Sample SQL Statement

The following SQL statement will return the total number of call attempts, and calls made and received by operators.

No distinction is made about the direction or the name of the operator group.

```
SELECT
    COUNT(*)
FROM
    session
WHERE
    (ServiceTypeID1 = 0 AND ExtensionID1 IS NOT NULL)
    OR (ServiceTypeID2 = 0 AND ExtensionID2 IS NOT NULL)
```

5.6 Calls Made to and from Automated Attendants

5.6.1 References

Table(s): session

Field(s):

- ExtensionID1
- ServiceTypeID1
- ServiceTypeID2

5.6.2 Sample SQL Statement

The following SQL statement will return the total number of call attempts, and calls made to all automated attendants.

No distinction is made about the name of the automated attendant.

```
SELECT
    COUNT(*)
FROM
    session
WHERE
    (ServiceTypeID1 = 2 AND ExtensionID1 IS NOT NULL)
    OR (ServiceTypeID2 = 2 AND ExtensionID2 IS NOT NULL)
```

Database Access to Call Detail Records

MX Enterprise Media Exchange

5.7 Called Party and Dial Plan used for Calls Made by an Individual User

5.7.1 References

Table(s): extension, session

Field(s):

- extension.ExtensionID
- extension.Extension
- session.SessionID
- session.ExtensionID1
- session.DialPlanName

5.7.2 Sample SQL Statement

The following SQL statement will return the called party number and dial plan entry used for all calls and call attempts from a individual user.

The user in this example has a configured extension of 404.

```
SELECT FinallyCalledParty, DialPlanName
FROM
    session, extension
WHERE
    extension.ExtensionID = session.ExtensionID1
    AND ServiceExtension1 IS NULL
    AND extension.Extension LIKE '404'
```

5.8 Emergency Calls and Attempts

5.8.1 References

Table(s): session, mxuser

Field(s):

- session.IsEmergencyCall
- session.ExtensionID1
- mxuser.ExtensionID
- mxuser.FirstName
- mxuser.LastName

5.8.2 Sample SQL Statement

The following SQL statement will return the first and last name of all users who made or attempted to make an emergency call.

```
SELECT FirstName, LastName
FROM
```

Database Access to Call Detail Records

MX Enterprise Media Exchange

```
    session, mxuser
WHERE
    session.ExtensionID1 = mxuser.ExtensionID
    AND session.IsEmergencyCall = 1
```

5.9 Call and Call Attempts from a User Extension

5.9.1 References

Table(s): extension, session

Field(s):

```
    extension.ExtensionID
    extension.Extension
    session.ExtensionID1
    session.ServiceExtension1
```

5.9.2 Sample SQL Statement

The following SQL statement will return the total number of calls and call attempts made from a user's extension.

The user in this example has a configured extension of 404.

```
SELECT (*)
FROM
    session, extension
WHERE
    extension.ExtensionID = session.ExtensionID1
    AND ServiceExtension1 IS NULL
    AND extension.Extension LIKE '404'
```

5.10 Total number of Internal Calls

5.10.1 References

Table(s): session

Field(s):

```
    LegType1
    LegType2
    InternalPartyType1
    InternalPartyType2
```

Database Access to Call Detail Records

MX Enterprise Media Exchange

5.10.2 Sample SQL Statement

The following SQL statement is used to return the total number of internal calls and internal call attempts.

Calls to system services such as operator, ACD and automated attendant are excluded from this example.

```
SELECT
    COUNT(*)
FROM
    session
WHERE
    LegType1 = 1 AND LegType2 = 1
    AND InternalPartyType1 = 0 AND InternalPartyType2 = 0
```

5.11 Total Number of Inbound Calls

5.11.1 References

Table(s): session

Field(s):

LegType1
LegType2
InternalPartyType2

5.11.2 Sample SQL Statement

The following SQL statement is used to return the total number of inbound calls for received by users only. Calls to system services such as operator, ACD and automated attendant are excluded from the report.

```
SELECT
    COUNT(*)
FROM
    session
WHERE
    LegType1 = 2 AND LegType2 = 1 AND InternalPartyType2 = 0
```

5.12 Total Number of Outbound Calls and Call Attempts

5.12.1 References

Table(s): session

Database Access to Call Detail Records

MX Enterprise Media Exchange

Field(s):

LegType1
LegType2
InternalPartyType1

5.12.2 Sample SQL Statement

The following SQL statement is used to return the total number of outbound calls and outbound call attempts.

Calls to system services such as operator, ACD and automated attendant are excluded from the report.

```
SELECT
    COUNT(*)
FROM
    session
WHERE
    LegType1 = 1 AND LegType2 = 2 AND InternalPartyType1 = 0
```

5.13 The Ten Longest Calls Made on the System

5.13.1 References

Table(s): session

Field(s):

DisconnectTimeStamp
ConnectTimeStamp

5.13.2 Sample SQL Statement

The following SQL statement is used to return the 10 longest calls made on the system. Their is not distinction made between internal, inbound, or outbound calls.

```
SELECT
    sec_to_time(time_to_sec(DisconnectTimeStamp) -
    time_to_sec(ConnectTimeStamp))
    AS len
FROM session
WHERE
    ConnectTimeStamp > '1970/01/02' ORDER BY len limit 10
```

Database Access to Call Detail Records

MX Enterprise Media Exchange

5.14 Total Answered Calls Made by a User for a Specified Date Range

5.14.1 References

Table(s): extension, session

Field(s):

- extension.ExtensionID
- extension.Extension
- session.ExtensionID1
- session.ServiceExtension1
- session.ConnectTimeStamp
- session.DisconnectTimeStamp

5.14.2 Sample SQL Statement

The following SQL statement is used to determine the total number of calls only (call attempts will be excluded) for the time period of January 01, 2004 and May 01, 2004.

The user in this example has a configured extension of 404.

```
SELECT (*)
FROM
    session, extension
WHERE
    extension.ExtensionID = session.ExtensionID1
    AND ServiceExtension1 IS NULL
    AND extension.Extension LIKE '404'
    AND ConnectTimeStamp > '1970/01/02'
    AND ConnectTimeStamp BETWEEN '2004/01/01' AND '2004/05/01'
```

6 Sample CDR Generation

6.1 Visual Basic.NET

Visual Basic.NET is the latest version of Visual Basic from Microsoft. You can easily create a Windows GUI that displays the CDR data.

6.1.1 ODBC.NET – MyODBC Architecture

A diagram of the ODBC.NET – MyODBC Architecture is shown in figure 16

Setting up the ODBC.NET Environment

1. Download and install the latest .NET Framework SDK

Database Access to Call Detail Records

MX Enterprise Media Exchange

<http://msdn.microsoft.com/netframework/downloads/updates/default.aspx>

2. Install Microsoft Data Access Components (MDAC) 2.6 or later. MDAC 2.7 is recommended

<http://msdn.microsoft.com/data/>

3. Install the ODBC.NET Provider

<http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=6CCD8427-1017-4F33-A062-D165078E32B1>

4. Setup an MyODBC DSN to be used for connecting to MySQL by following steps described in section 6.1.2 on page 23.

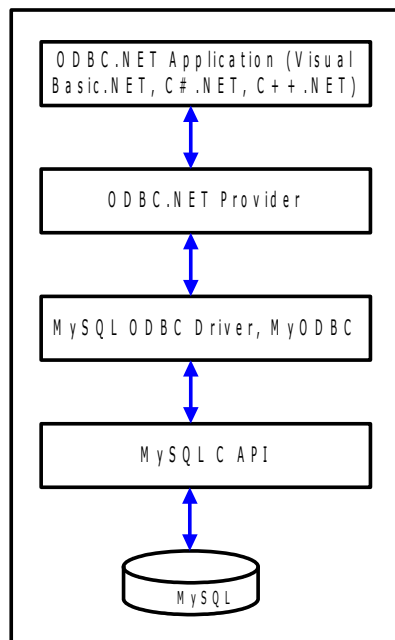


Figure 16 ODBC.NET – MyODBC Architecture

6.1.2 Adding a Data Source on Windows

Perform the following steps to add a data source to Windows that will be used to connect to the CDR database.

1. Open the **Data Sources (ODBC)** accessed from **Control Panel | Administrative Tools**. A window as shown in figure 17 should be displayed.
2. Click the **Add** button and choose **MySQL ODBC 3.51 Driver** as the driver you want to use to setup the data source.

Database Access to Call Detail Records

MX Enterprise Media Exchange

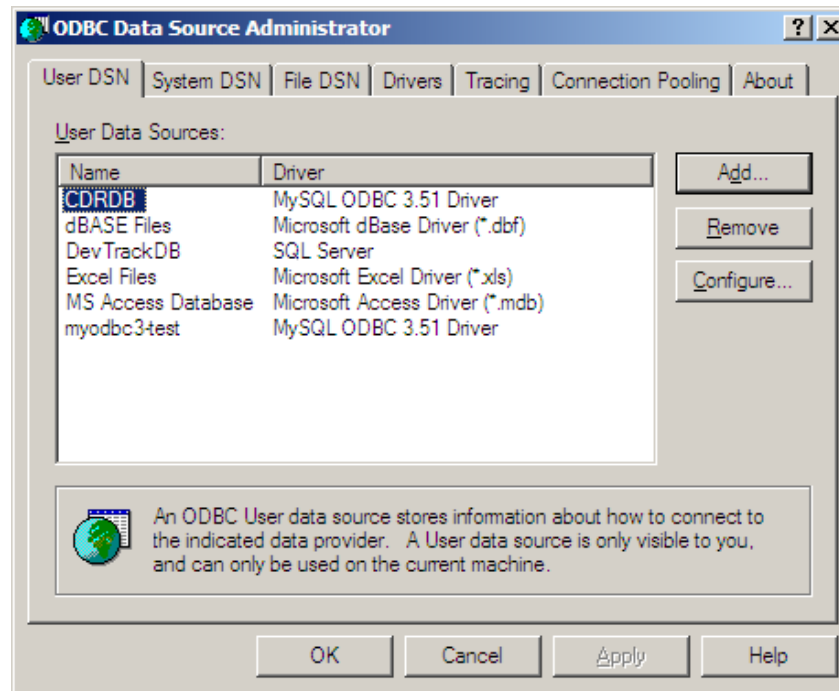


Figure 17 ODBC Data Source Administrator

3. A window as shown in figure 18 is displayed with the proper settings.
 - Data Source Name.* Name for the data source that will be used by calling program
 - Description.* Any text string to describe the data source
 - Host/Server Name (or IP).* The IP Address of the MX system
 - Database Name.* cdrdb
 - User.* cdr_reader
 - Password.* cdr_reader
 - Port.* 3306
 - SQL command on connect.* Not required to configure
4. Click the **OK** button
5. The data source should be tested, to test, click the **Test Data Source** button. A message box as shown in figure 19 will be displayed if the data source is properly configured. If there is an error in the data source configuration, a message box as shown in figure 20 will be displayed. If you receive an error, check all your settings and verify that the MX system is operational and then retest the data source.

Database Access to Call Detail Records

MX Enterprise Media Exchange

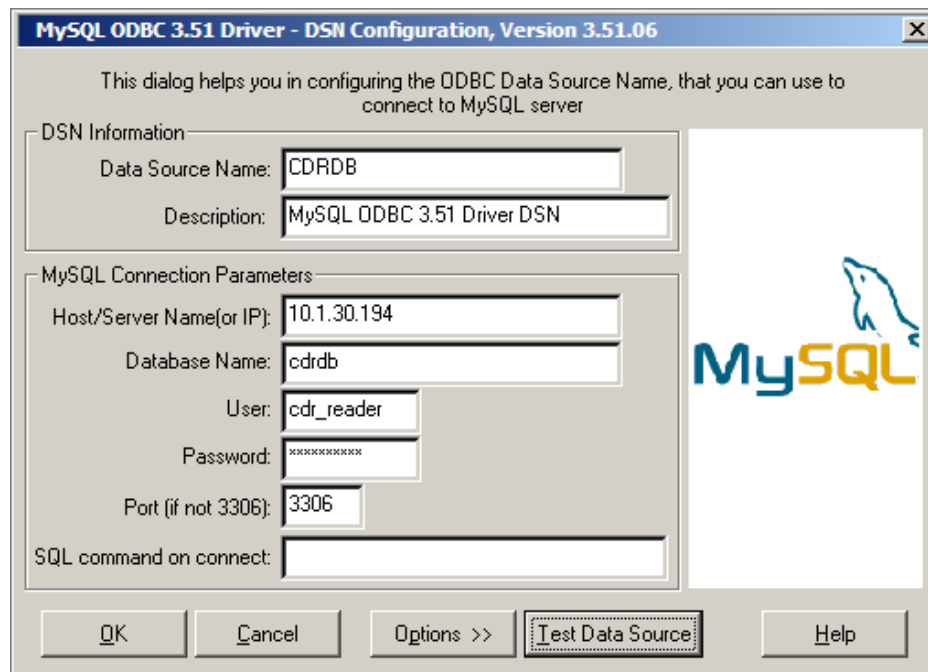


Figure 18 MySQL ODBC Settings to Access MX CDR Database

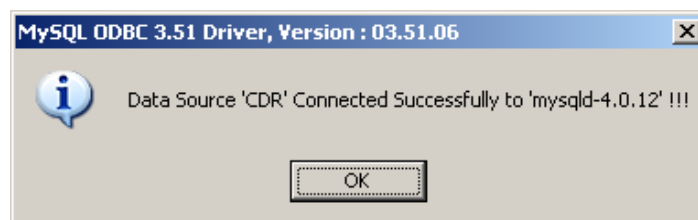


Figure 19 Successfully Configured Data Source

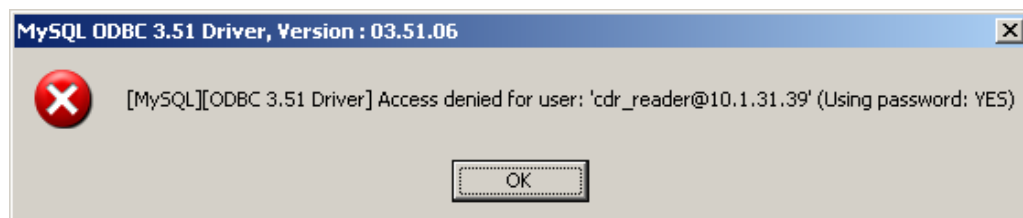


Figure 20 Unsuccessful Configured Data Source

Database Access to Call Detail Records

MX Enterprise Media Exchange

6.1.3 Setting up Visual Basic.NET to Use the ODBC.NET Provider

Perform the following steps to add the ODBC.NET provider to the Visual Basic.NET environment.

1. Create a standard Windows application.
2. Add the ODBC.NET reference to Visual Basic by right clicking on the project and selecting the **Add Reference** menu item as shown in figure 21.
3. A window as shown in figure 22 will be displayed. Double click the **Microsoft.Data.Odbc.dll** component from the .NET tab and click the **OK** button.
4. The Microsoft.Data.Odbc reference should appear in the references section of the Visual Basic project as shown in figure 23.
5. Use the **imports** statement to add a reference to the database driver before any other program declarations.

```
Imports Microsoft.Data.Odbc
Public Class Main
...code
...code
End Class
```

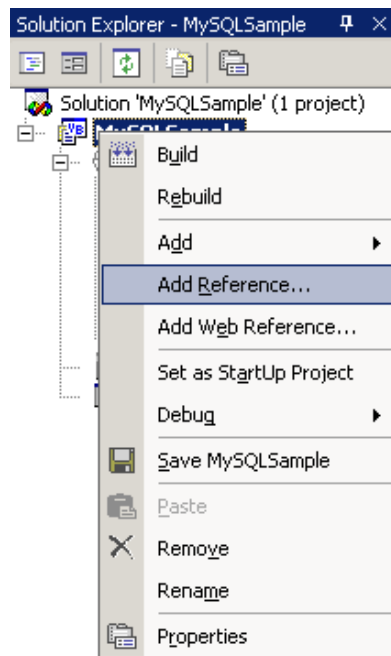


Figure 21 Adding the ODBC.NET Reference to Visual Basic

Database Access to Call Detail Records

MX Enterprise Media Exchange

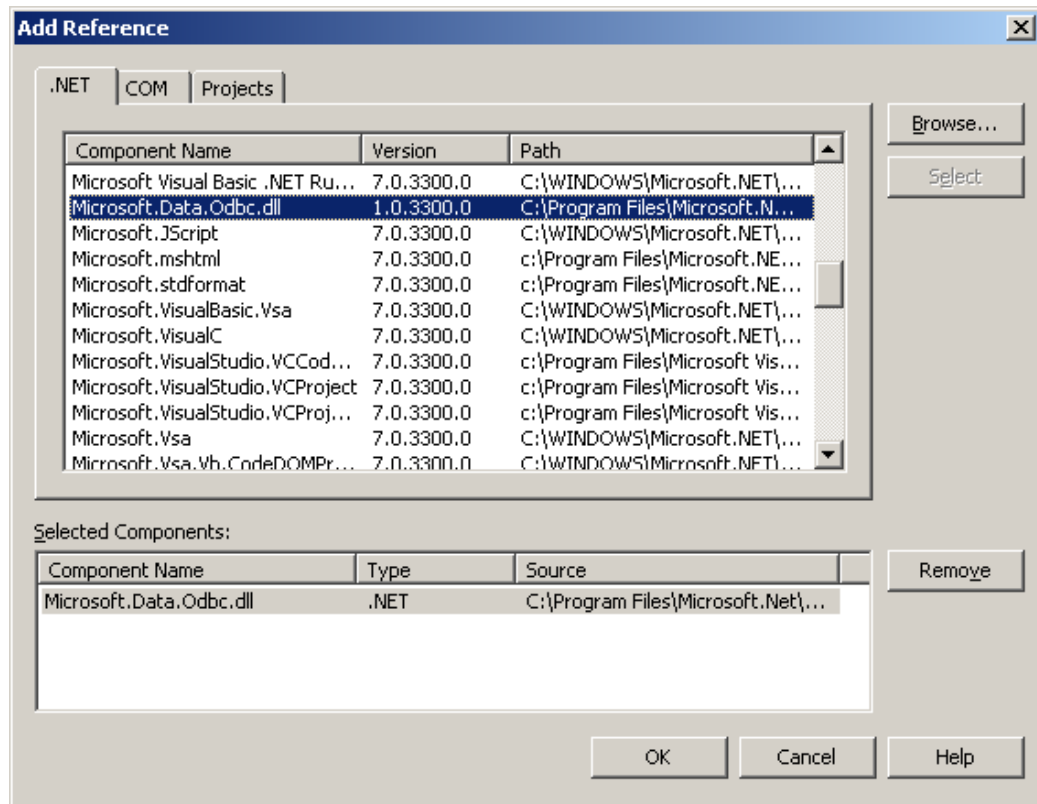


Figure 22 Adding a Reference to Visual Basic

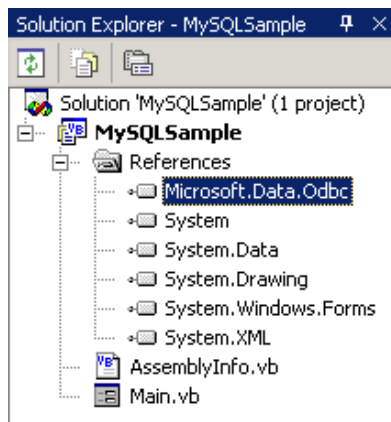


Figure 23 Microsoft.Data.Odbc Reference

Database Access to Call Detail Records

MX Enterprise Media Exchange

6.1.4 Connecting to the Database

A connection to the database is required to execute the database functions. An example of how to make a database connection with the DSN that was created from section 6.1.2 on page 23. The following example places the connection procedure in a class named CData.vb.

```
Imports Microsoft.Data.Odbc

Public Class CData

    'Class for database connection and DataReader object

    Dim m_cn As New OdbcConnection()

    Public Sub New()

        Try

            m_cn.ConnectionString() = "DSN=CDRDB"

            m_cn.Open()

        Catch ex As Exception

            MsgBox(ex.Message, MsgBoxStyle.Critical, "MySQL VB.NET

Sample")

        End Try

    End Sub

    Public Function GetReader(ByVal sSql As String) As OdbcDataReader

        'Returns a datareader object for a given SQL expression

        'A datareader is used to return data from the database. Since read-

only

        'access is allowed, the datareader can be used to hold the returned

data

        'from the executed SQL statements

        Try

            Dim cmd As OdbcCommand = m_cn.CreateCommand()

            cmd.CommandText = sSql

            Dim rdr As OdbcDataReader = cmd.ExecuteReader()

            Return rdr

        Catch ex As Exception

            MsgBox(ex.Message, MsgBoxStyle.Critical, "MySQL VB.NET

Sample")

        End Try

    End Function

End Class
```

Database Access to Call Detail Records

MX Enterprise Media Exchange

6.1.5 Sample Application

The sample application is intended to demonstrate how to make a database connection and execute a simple SQL query to generate a basic report.

The sample application will use the datareader object to hold the data that is returned from the query results. A combo box will be populated with all user extensions and the **Called Party Number** and the **Call Duration** will be displayed in a listview control for the selected user extension.

The following procedure is used to load the user extensions to the combo box.

```
Private Sub LoadExtensions()
    Dim AnItem As ExtraData
    'The AnItem is a reference to a subclassed combo box to hold the data
    'for the EXTENSIONID. The code will be displayed, but does not relate
    'to usng the database functions.
    Dim rdr As OdbcDataReader
    rdr = CDataServices.GetReader("SELECT EXTENSION, EXTENSIONID FROM
extension")
    'The query result is stored in the rdr datareader object

    While rdr.Read
        'The datareader will be used to populate the combo box
        cboExtensions.Items.Add(New
ExtraData(CStr(rdr.Item("EXTENSION")),
        CStr(rdr.Item("EXTENSIONID"))))
    End While

    If cboExtensions.Items.Count = 0 Then
        cboExtensions.Items.Add("No Extension Records")
        cboExtensions.SelectedIndex = 0
    Else
        cboExtensions.SelectedIndex = 0
    End If

    rdr.Close()
    m_cn.Close()
End Sub
```

The code for the subclassed combo box

```
Class ExtraData
```

Database Access to Call Detail Records

MX Enterprise Media Exchange

```

'Create a subclass to store the EXTENSIONID with the EXTENSION data
in the
'combo box
Inherits ListViewItem

Public MyComboText As String
Public MyExtraData As String

Sub New(ByVal ShowText As String, ByVal Strng As String)
    MyBase.New()
    'transfer all incoming parameters to your local storage
    MyComboText = ShowText
    MyExtraData = Strng

    Me.Text = MyComboText
End Sub
End Class

```

The following procedure is used to load the listview with the **Called Party Number** and **Call Duration**.

```

Private Sub LoadCalls()
    'clear the list view before updating with new call records
    lvUsers.Items.Clear()
    'display the extension and the extension id
    'declare a copy of the subclass

    Dim AnItem As ExtraData
    'cast the incoming node into your subclass
    AnItem = CType(cboExtensions.SelectedItem, ExtraData)

    Dim rdr As OdbcDataReader
    rdr = CDataServices.GetReader("SELECT FINALLYCALLEDPARTYNO," & _
        CONNECTTIMESTAMP, DISCONNECTTIMESTAMP FROM session " & _
        "WHERE CONNECTTIMESTAMP '1970/01/02'> EXTENSIONID1 = '" &
        AnItem.MyExtraData & "'")

```

Database Access to Call Detail Records

MX Enterprise Media Exchange

'The query result is stored in the rdr datareader object

'Most the following code is related to formating the duration from seconds to

```
'minutes and seconds.
Dim StartCallTime As Date
Dim EndCallTime As Date
Dim CallDuration As Integer 'in seconds
Dim i As Integer = 0 'used to keep track of the number of records and
to remove
    call durations with 0 time
While rdr.Read
    lvUsers.Items.Add(CStr(rdr.Item("FINALLYCALLEDPARTYNO")))
    StartCallTime = rdr.Item("CONNECTTIMESTAMP")
    EndCallTime = rdr.Item("DISCONNECTTIMESTAMP")
    CallDuration = DateDiff(DateInterval.Second, StartCallTime,
    EndCallTime)

    Dim DayFormat As String
    Dim HourFormat As String
    Dim MinuteFormat As String
    Dim SecondFormat As String

    If CallDuration = 0 Then
        'remove entry that have a call duration of 0 seconds. If you
want to
        'display call durations with 0 seconds, this code is not
required
        lvUsers.Items.RemoveAt(i)
    Else
        'Convert the seconds to minutes and seconds. Further
code would
        be required to add support for hours, days, etc.
        If CallDuration > 59 Then
            Dim IntegerResult As Integer
            Dim RemainderResult As Integer
```

Database Access to Call Detail Records

MX Enterprise Media Exchange

```

IntegerResult = CallDuration \ 60
RemainderResult = CallDuration Mod 60
'Format the minute and second stings
If IntegerResult = 1 Then
    MinuteFormat = CStr(IntegerResult & "
minute")
Else
    MinuteFormat = CStr(IntegerResult & "
minutes")
End If

If RemainderResult = 1 Then
    SecondFormat = CStr(RemainderResult & "
second")
Else
    SecondFormat = CStr(RemainderResult & "
seconds")
End If

lvUsers.Items(i).SubItems.Add(MinuteFormat & "
and " &
    SecondFormat)
Else
    If CallDuration = 1 Then
lvUsers.Items(i).SubItems.Add(CStr(CallDuration)
    & " second")
    Else
lvUsers.Items(i).SubItems.Add(CStr(CallDuration)
    & " seconds")
    End If
    End If
    i = i + 1
'Display the total number of call records for extension
lblTotalCalls.Visible = True
lblTotalCalls.Text = "Total calls made for extension " &

```


Database Access to Call Detail Records

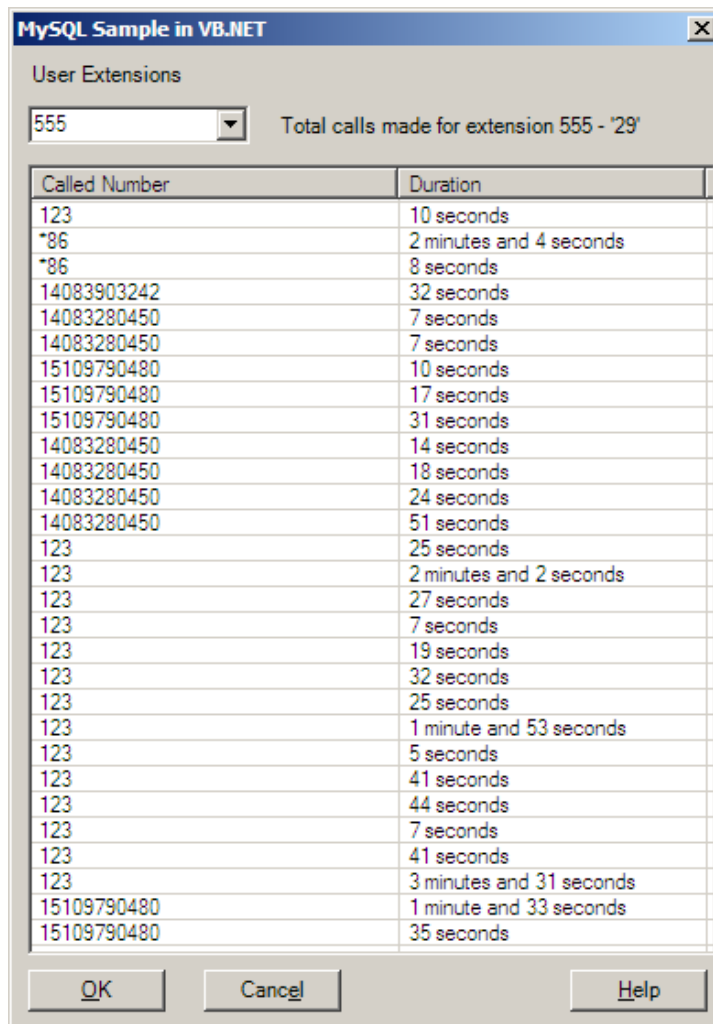
MX Enterprise Media Exchange

```

        cboExtensions.Text & " - " & "'" & i.ToString & "'"
    End If
End While
rdr.Close()
m_cn.Close()
End Sub

```

A sample screen shot of the application is shown in figure 24.



Called Number	Duration
123	10 seconds
*86	2 minutes and 4 seconds
*86	8 seconds
14083903242	32 seconds
14083280450	7 seconds
14083280450	7 seconds
15109790480	10 seconds
15109790480	17 seconds
15109790480	31 seconds
14083280450	14 seconds
14083280450	18 seconds
14083280450	24 seconds
14083280450	51 seconds
123	25 seconds
123	2 minutes and 2 seconds
123	27 seconds
123	7 seconds
123	19 seconds
123	32 seconds
123	25 seconds
123	1 minute and 53 seconds
123	5 seconds
123	41 seconds
123	44 seconds
123	7 seconds
123	41 seconds
123	3 minutes and 31 seconds
15109790480	1 minute and 33 seconds
15109790480	35 seconds

Figure 24 Sample Application Output