IP Office 8.1

IP Office Customer Call Reporter
Custom Reporting

- Issue 02a - (07 June 2012)

Contents

Chapter 1. Overview

1. Overview

This document can be used by third party developers as a reference when designing and writing an application that can generate reports using data mined from the IP Office Customer Call Reporter database. This document provides information on how to connect to the IP Office Customer Call Reporter database, discusses the IP Office Customer Call Reporter database design and provides a description of the data stored in the IP Office Customer Call Reporter database.

The developer using this information is deemed to have the knowledge required to access and retrieve data from MS SQL.

The information in this document can be used to create custom reports for IP Office Customer Call Reporter 8.1.

- **! WARNING**
  The design of custom reports using the data provided in the IP Office Customer Call Reporter database is the property of the designer and all support associated with it is to be provided by that designer. Removing or modifying any of the Tables, Relations, Stored Procedures, Functions or data within the database will affect IP Office Customer Call Reporter operation and is not supported by Avaya. The existing Stored Procedures and Functions that are part of the database can only be used on an 'as is' basis. New Stored Procedures and Functions can be added, however this should only be done by users with MS-SQL experience and should be tested and validated by them before being applied to a customer system. Avaya will not provide any support for third-party Stored Procedures and Functions.

1.1 Database Access

The IP Office Customer Call Reporter Catalog is called AvayaSBCCRT.

The developer should either get the account and password to use from the administrator that installed IP Office Customer Call Reporter or even better, an account should be created for the developer with just enough privileges to satisfy the requirements. The specific account can also be useful when diagnosing issues with the database by being able to track which applications (IP Office Customer Call Reporter or the Custom Report application) had database transactions.

The connection string for SQL Express needs the default instance name appended to the hostname or IP address (e.g. DataSource=localhost\SQLEXPRESS;).

A backup of the database should be taken as the account used can have the capability to alter the database in such ways that IP Office Customer Call Reporter could become inoperative.

Sample C# code to connect to the database:

```
SqlConnection connection = new SqlConnection("Data Source=localhost\\SQLEXPRESS;Initial
Catalog=AvayaSBCCRT;uid=username;pwd=password");
```

1.2 Remote Access

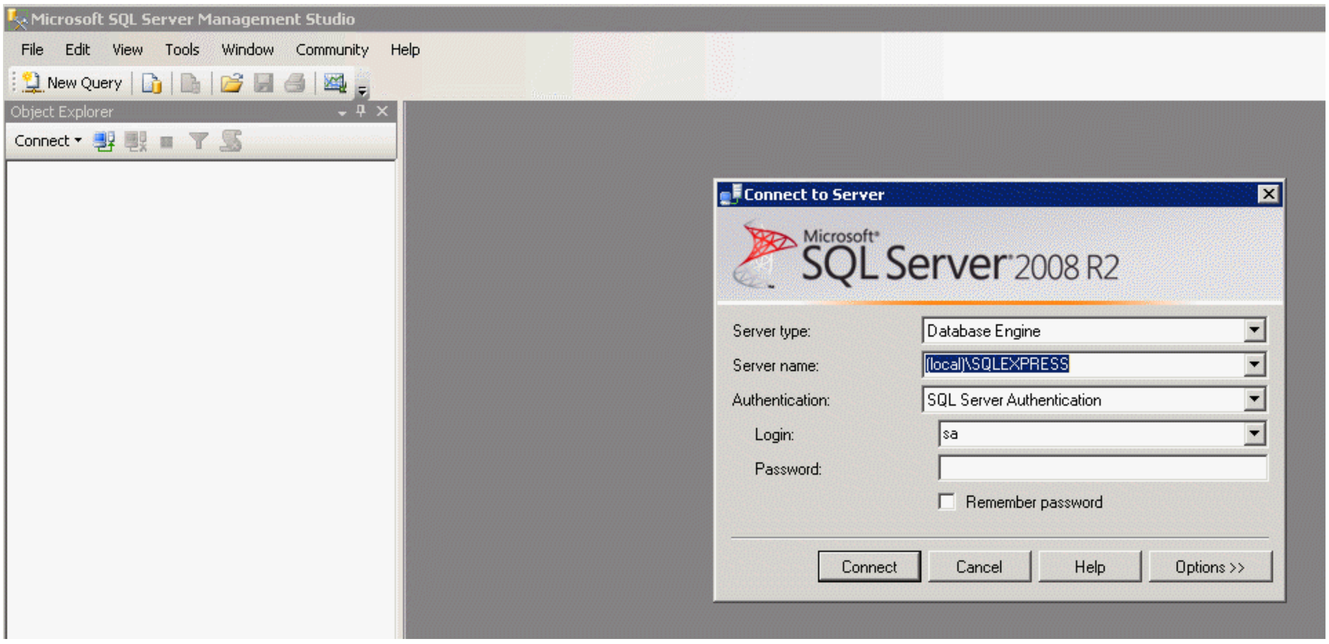
If remote access to the database is needed, certain TCP/IP protocols and the SQL browser service need to be enabled on the SQL Server PC. In addition, firewall rules may need to be modified. This is described in the document, http://support.microsoft.com/kb/914277 and is not needed for local access which is preferred.

1.3 Management Studio

The IP Office Customer Call Reporter database can be "viewed" using the Management Studio application. This can be obtained for free http://www.microsoft.com/downloads/details.aspx?FamilyID=08E52AC2-1D62-45F6-9A4A-4B76A8564A2B&displaylang=en.

This tool will show the database and the relation between tables. It will also show the definition for each field in the table and the Stored Procedures and Functions that can be used by the developers if needed. A few screen captures below explains how to use this tool to understand the IP Office Customer Call Reporter database.

- **! WARNING**
  The design of custom reports using the data provided in the IP Office Customer Call Reporter database is the property of the designer and all support associated with it is to be provided by that designer. Removing or modifying any of the Tables, Relations, Stored Procedures, Functions or data within the database will affect IP Office Customer Call Reporter operation and is not supported by Avaya. The existing Stored Procedures and Functions that are part of the database can only be used on an 'as is' basis. New Stored Procedures and Functions can be added, however this should only be done by users with MS-SQL experience and should be tested and validated by them before being applied to a customer system. Avaya will not provide any support for third-party Stored Procedures and Functions.

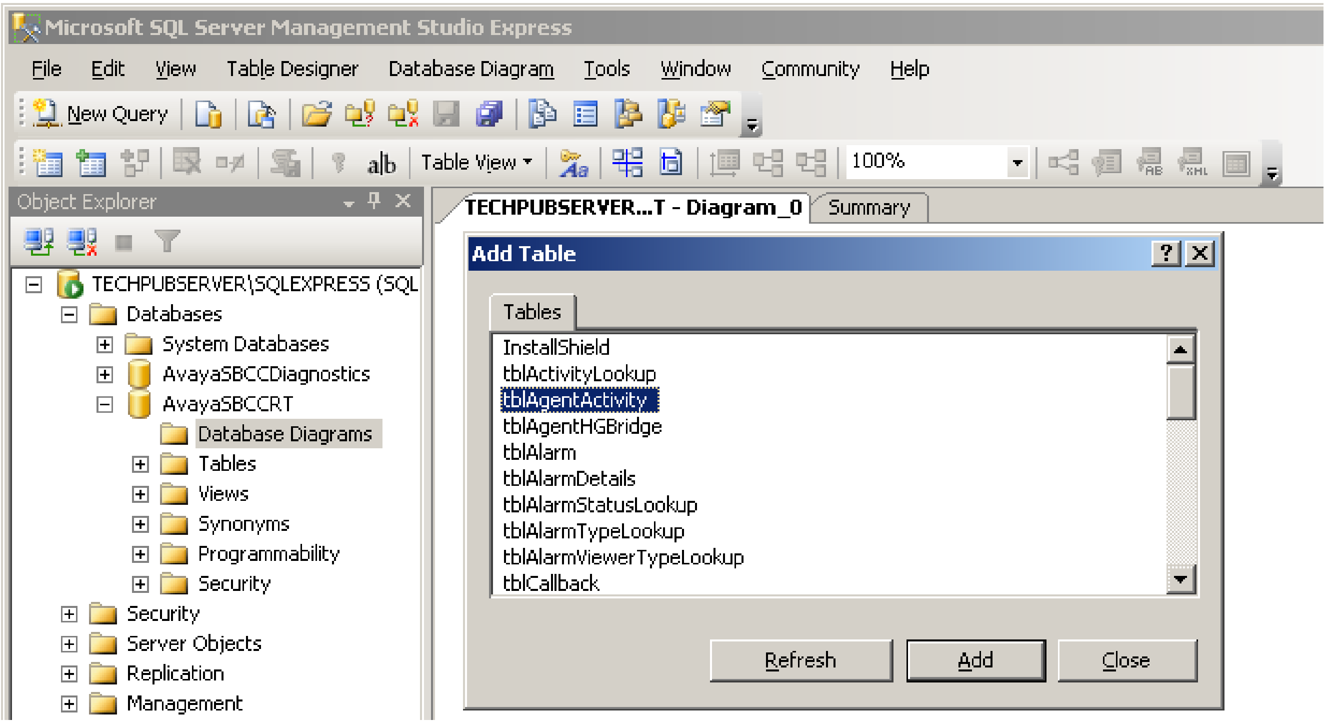

Microsoft SQL Server Management Studio
File Edit View Tools Window Community Help
New Query
Object Explorer
Connect
Connect to Server
Microsoft SQL Server 2008 R2
Remember password
Connect Cancel Help Options >>

Server type: Database Engine
Server name: (local)\SQLEXPRESS
Authentication: SQL Server Authentication
Login: sa
Password:

Database Diagrams

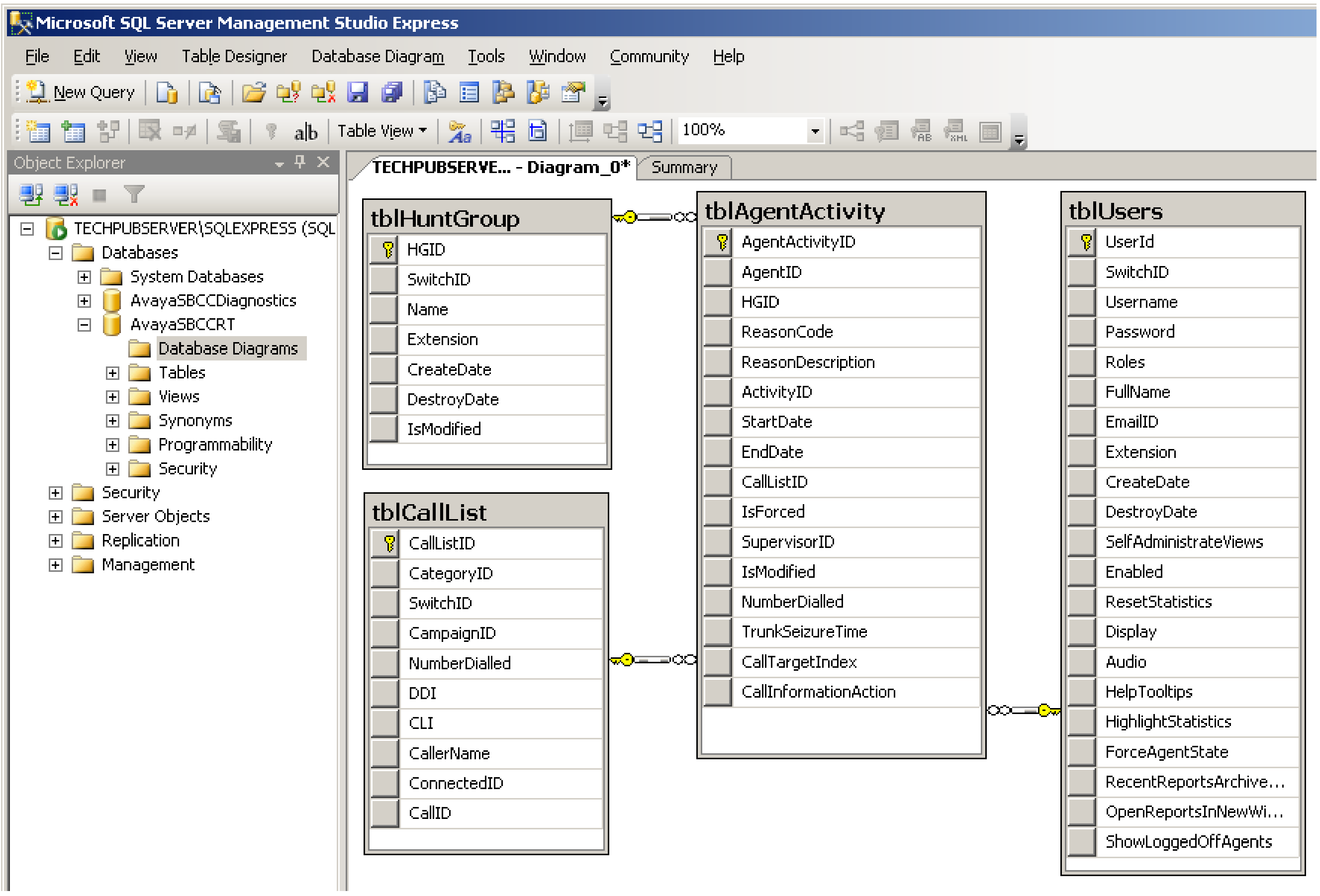
The tool can be used to display the database schema with the relationship between the tables. It is important to note that this is not a passive view, changes made to the diagram can affect connections within the database.

First, add a table to the Diagram pane.



InstallShield
tblActivityLookup
tblAgentActivity
tblAgentHGBridge
tblAlarm
tblAlarmDetails
tblAlarmStatusLookup
tblAlarmTypeLookup
tblAlarmViewerTypeLookup
tblCallback

Then, a right click on the added table and a request to add related tables can be made. That will show the relationship between tables. Different views can be selected (table names only, with keys only, with column definitions, etc).



tblHuntGroup: HGID, SwitchID, Name, Extension, CreateDate, DestroyDate, IsModified

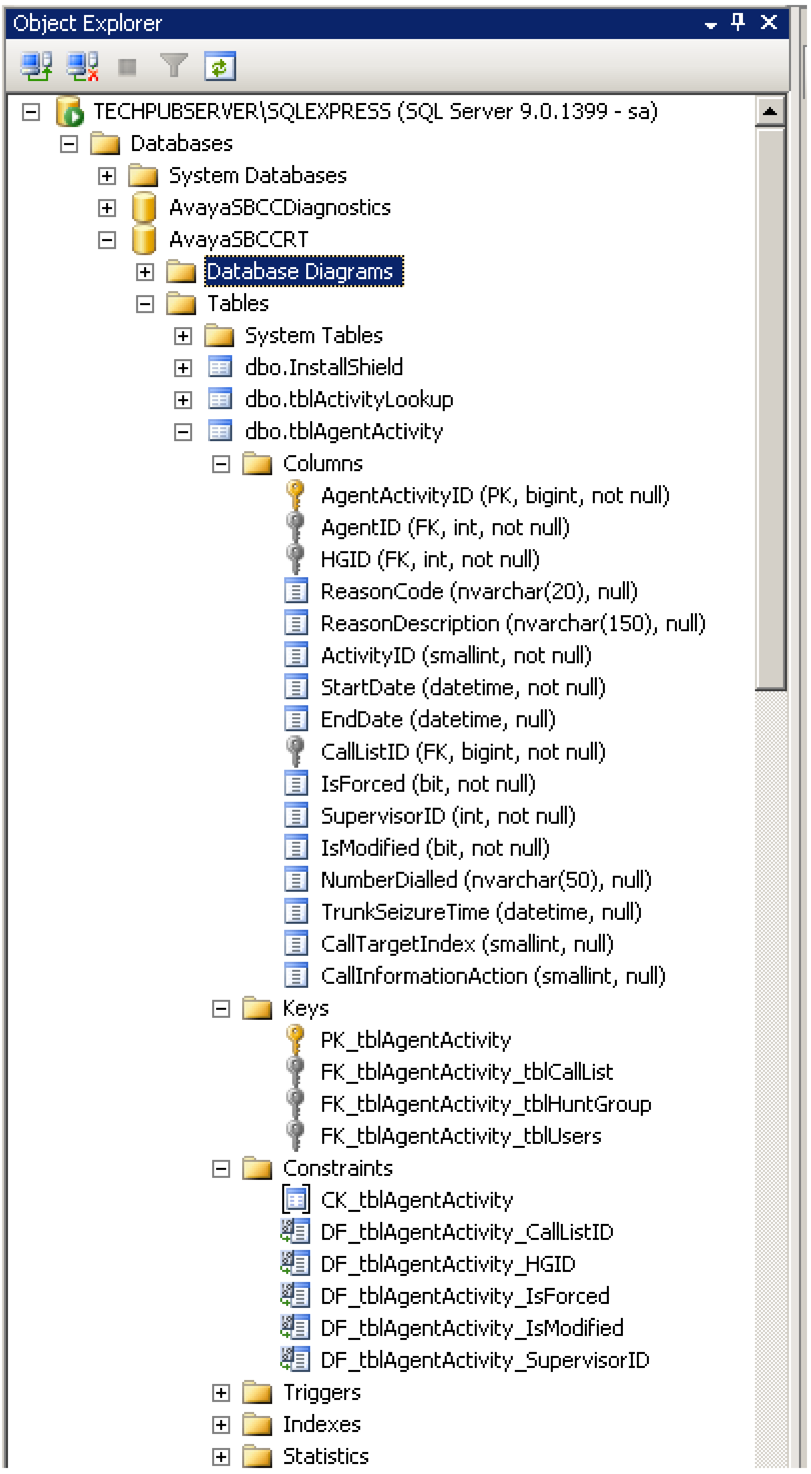
tblCallList: CallListID, CategoryID, SwitchID, CampaignID, NumberDialled, DDI, CLI, CallerName, ConnectedID, CallID

tblAgentActivity: AgentActivityID, AgentID, HGID, ReasonCode, ReasonDescription, ActivityID, StartDate, EndDate, CallListID, IsForced, SupervisorID, IsModified, NumberDialled, TrunkSeizureTime, CallTargetIndex, CallInformationAction

tblUsers: UserId, SwitchID, Username, Password, Roles, FullName, EmailID, Extension, CreateDate, DestroyDate, SelfAdministrateViews, Enabled, ResetStatistics, Display, Audio, HelpTooltips, HighlightStatistics, ForceAgentState, RecentReportsArchive..., OpenReportsInNewWi..., ShowLoggedOffAgents

Tables

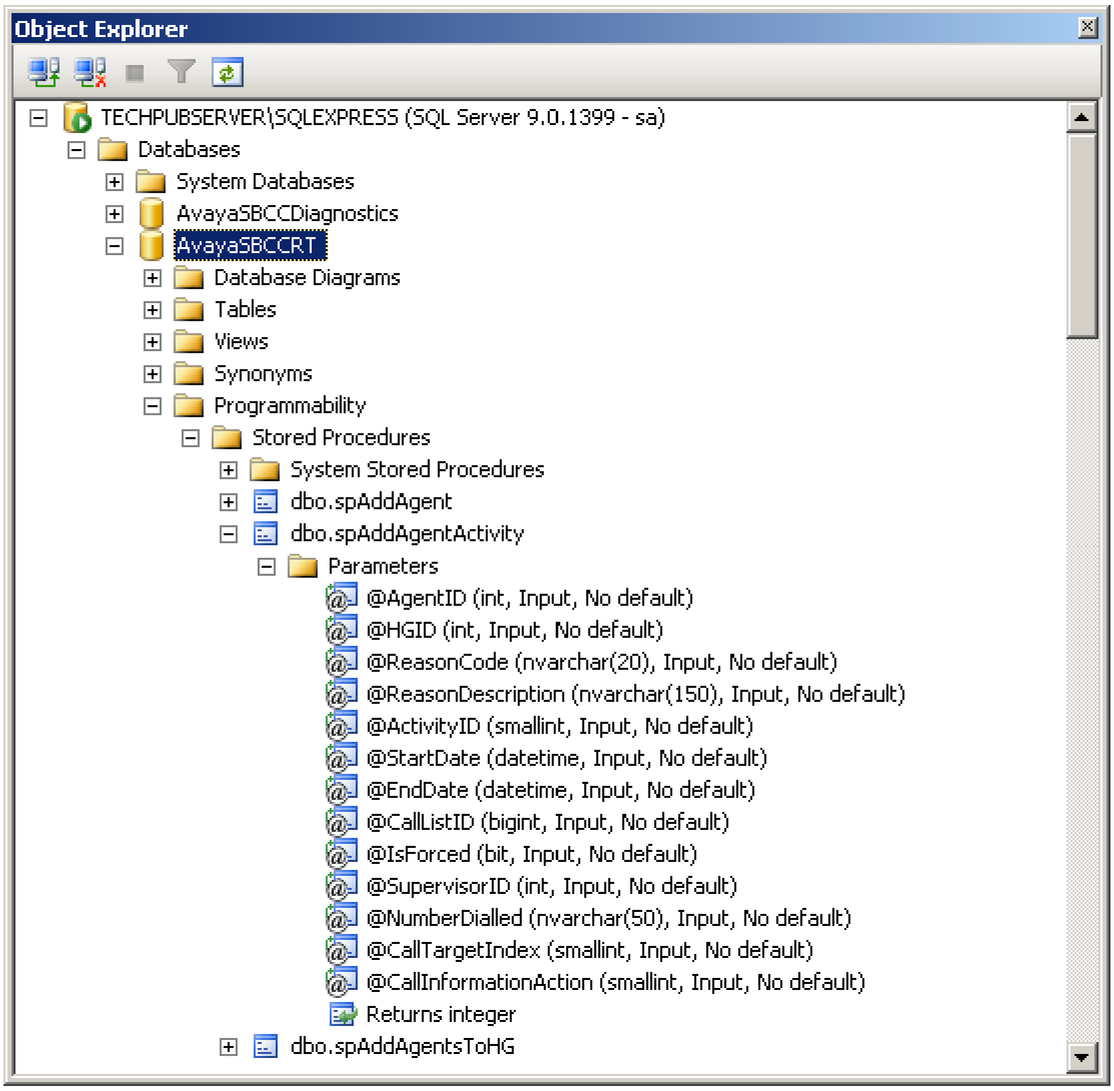
The Tables 12 section has definitions (Columns, Keys, Constraints, and so on) for each database table.



```
TECHPUBSERVER\SQLEXPRESS (SQL Server 9.0.1399 - sa)
  Databases
    System Databases
    AvayaSBCCDiagnostics
    AvayaSBCCRT
      Database Diagrams
      Tables
        System Tables
        dbo.InstallShield
        dbo.tblActivityLookup
        dbo.tblAgentActivity
          Columns
            AgentActivityID (PK, bigint, not null)
            AgentID (FK, int, not null)
            HGID (FK, int, not null)
            ReasonCode (nvarchar(20), null)
            ReasonDescription (nvarchar(150), null)
            ActivityID (smallint, not null)
            StartDate (datetime, not null)
            EndDate (datetime, null)
            CallListID (FK, bigint, not null)
            IsForced (bit, not null)
            SupervisorID (int, not null)
            IsModified (bit, not null)
            NumberDialled (nvarchar(50), null)
            TrunkSeizureTime (datetime, null)
            CallTargetIndex (smallint, null)
            CallInformationAction (smallint, null)
          Keys
            PK_tblAgentActivity
            FK_tblAgentActivity_tblCallList
            FK_tblAgentActivity_tblHuntGroup
            FK_tblAgentActivity_tblUsers
          Constraints
            CK_tblAgentActivity
            DF_tblAgentActivity_CallListID
            DF_tblAgentActivity_HGID
            DF_tblAgentActivity_IsForced
            DF_tblAgentActivity_IsModified
            DF_tblAgentActivity_SupervisorID
          Triggers
          Indexes
          Statistics
```

Stored Procedures

A list of the Stored Procedures 28 used by IP Office Customer Call Reporter can also be displayed.

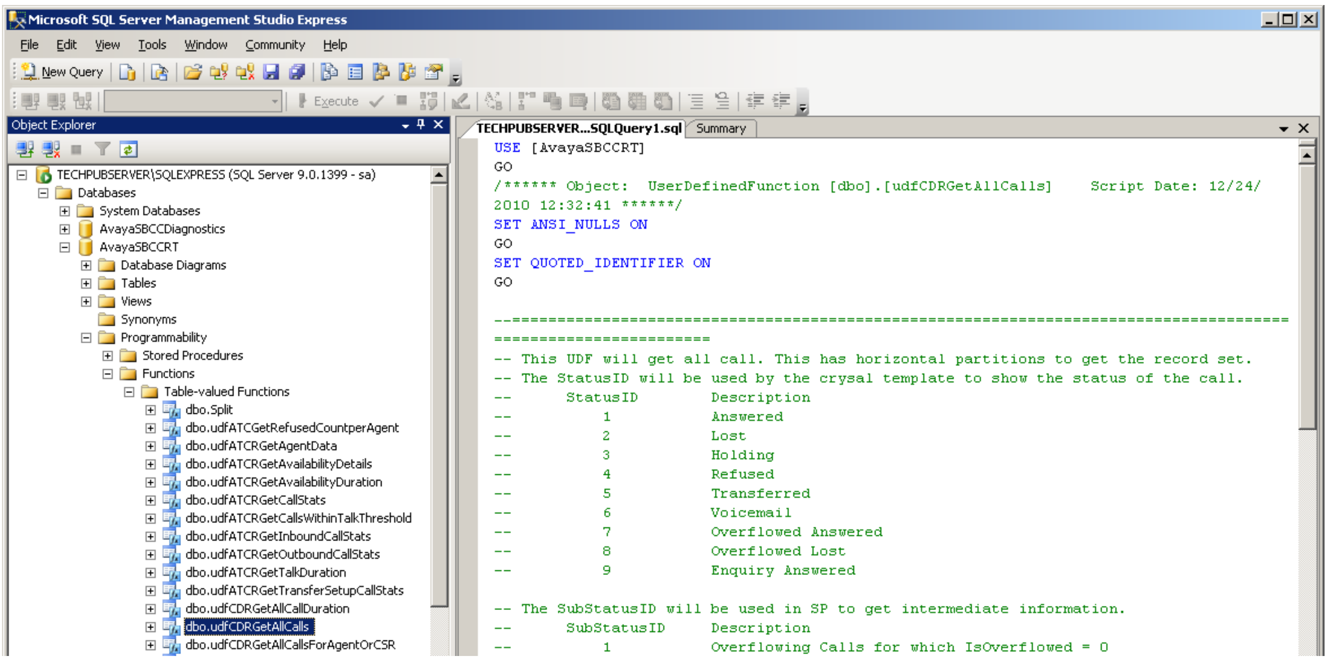


Object Explorer

```
TECHPUBSERVER\SQLEXPRESS (SQL Server 9.0.1399 - sa)
  Databases
    System Databases
    AvayaSBCCDiagnostics
    AvayaSBCCRT
      Database Diagrams
      Tables
      Views
      Synonyms
      Programmability
        Stored Procedures
          System Stored Procedures
          dbo.spAddAgent
          dbo.spAddAgentActivity
            Parameters
              @AgentID (int, Input, No default)
              @HGID (int, Input, No default)
              @ReasonCode (nvarchar(20), Input, No default)
              @ReasonDescription (nvarchar(150), Input, No default)
              @ActivityID (smallint, Input, No default)
              @StartDate (datetime, Input, No default)
              @EndDate (datetime, Input, No default)
              @CallListID (bigint, Input, No default)
              @IsForced (bit, Input, No default)
              @SupervisorID (int, Input, No default)
              @NumberDialled (nvarchar(50), Input, No default)
              @CallTargetIndex (smallint, Input, No default)
              @CallInformationAction (smallint, Input, No default)
              Returns integer
          dbo.spAddAgentsToHG
```

Functions

The list of Functions 31 (Table-valued or Scalar-valued) can be viewed.


Microsoft SQL Server Management Studio Express
File Edit View Tools Window Community Help
New Query
Execute
Object Explorer
TECHPUBSERVER\SQLEXPRESS (SQL Server 9.0.1399 - sa)
Databases
System Databases
AvayaSBCCDiagnostics
AvayaSBCCRT
Database Diagrams
Tables
Views
Synonyms
Programmability
Stored Procedures
Functions
Table-valued Functions
dbo.Split
dbo.udfATCGetRefusedCountperAgent
dbo.udfATCRGetAgentData
dbo.udfATCRGetAvailabilityDetails
dbo.udfATCRGetAvailabilityDuration
dbo.udfATCRGetCallStats
dbo.udfATCRGetCallsWithinTalkThreshold
dbo.udfATCRGetInboundCallStats
dbo.udfATCRGetOutboundCallStats
dbo.udfATCRGetTalkDuration
dbo.udfATCRGetTransferSetupCallStats
dbo.udfCDRGetAllCallDuration
dbo.udfCDRGetAllCalls
dbo.udfCDRGetAllCallsForAgentOrCSR
TECHPUBSERVER...SQLQuery1.sql
Summary

```
USE [AvayaSBCCRT]
GO
/****** Object:  UserDefinedFunction [dbo].[udfCDRGetAllCalls]    Script Date: 12/24/
2010 12:32:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

--==========================================================================================
=======================
-- This UDF will get all call. This has horizontal partitions to get the record set.
-- The StatusID will be used by the crysal template to show the status of the call.
--      StatusID        Description
--          1           Answered
--          2           Lost
--          3           Holding
--          4           Refused
--          5           Transferred
--          6           Voicemail
--          7           Overflowed Answered
--          8           Overflowed Lost
--          9           Enquiry Answered

-- The SubStatusID will be used in SP to get intermediate information.
--      SubStatusID     Description
--          1           Overflowing Calls for which IsOverflowed = 0
```

Chapter 2.
Database Details

2. Database Details

- **! WARNING**
  The design of custom reports using the data provided in the IP Office Customer Call Reporter database is the property of the designer and all support associated with it is to be provided by that designer. Removing or modifying any of the Tables, Relations, Stored Procedures, Functions or data within the database will affect IP Office Customer Call Reporter operation and is not supported by Avaya. The existing Stored Procedures and Functions that are part of the database can only be used on an 'as is' basis. New Stored Procedures and Functions can be added, however this should only be done by users with MS-SQL experience and should be tested and validated by them before being applied to a customer system. Avaya will not provide any support for third-party Stored Procedures and Functions.

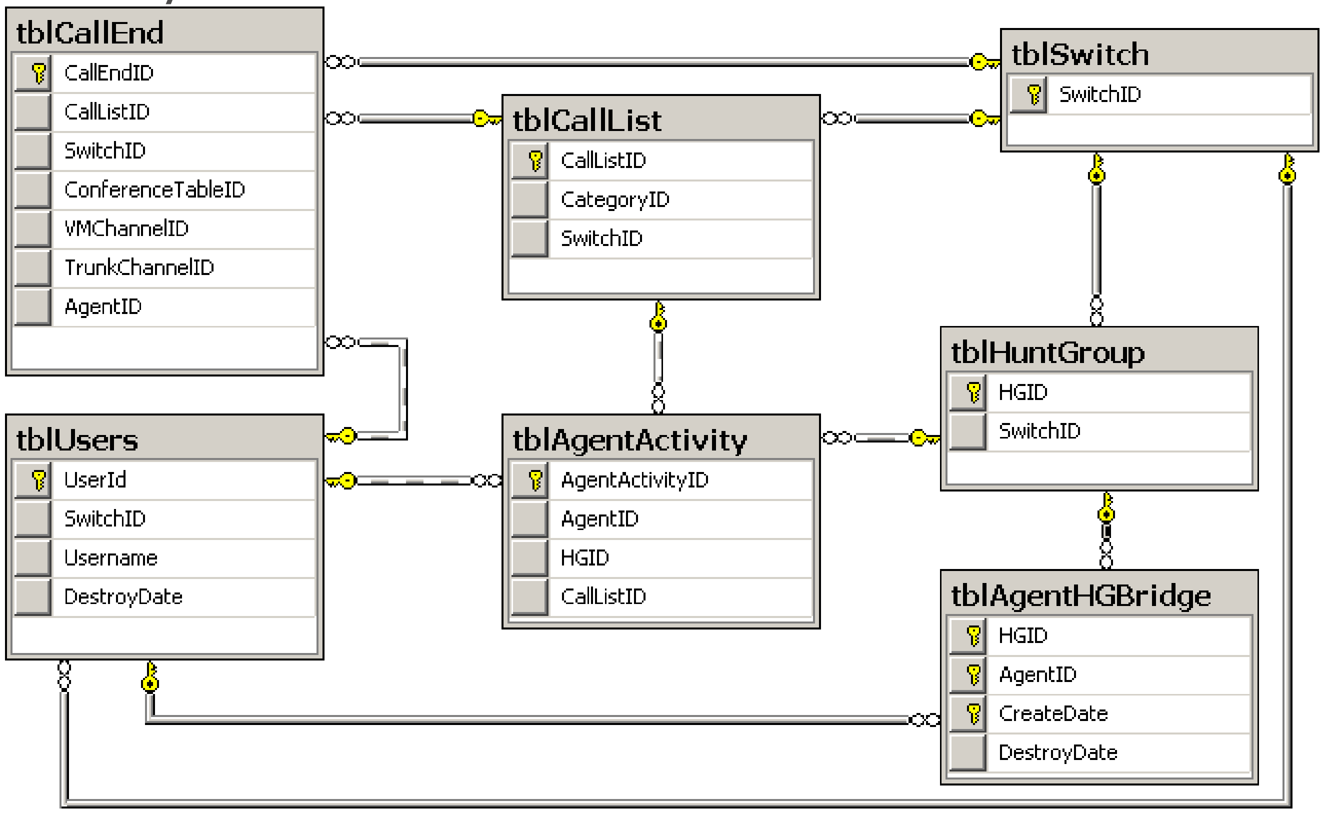
2.1 Database Tables

The tables described here are the ones used by IP Office Customer Call Reporter for reporting purposes.

Table	Category	Purpose
tblAgentActivity 14	**Transactional**	Contains detailed activity information for each Agent. It includes call related activity as well as non call activities like Login, Logout , Break , ACW, and so on for an Agent.
tblCallList 15	**Transactional**	Contains one record for each call. Call related information is stored here.
tblCallEnd 16	**Transactional**	Contains detailed state wise activity of a Call.
tblUsers 21	**Master**	Master table for Agents. This table also contains other user info like Supervisor, Administrator, etc.
tblHuntGroup 18	**Master**	List of Queue / Hunt group .
tblAgentHGBridge 14	**Intersect Master**	This table stores many to many relationship between Agent & Hunt group.
tblSwitch 21	**Master**	Contains one entry per IPOffice with details.
tblScheduledReport 19	**Scheduling**	List of the Reports Scheduled.
tblScheduledReportPeriodLookup 19	**Lookup**	Stores list of available Report Period options (e.g. Daily, Weekly).
tblScheduledReportFormatLookup 19	**Lookup**	Stores list of available Report Export options (e.g. PDF, Excel, etc.).
tblReportParameters 18	**Transactional**	Contains report parameters and other info for scheduled reports.
tblReports 19	**Master**	Stores list of the built in basic reports.
tblReportParametersScheduleLookup 18	**Lookup**	Stores the list of Schedule options.

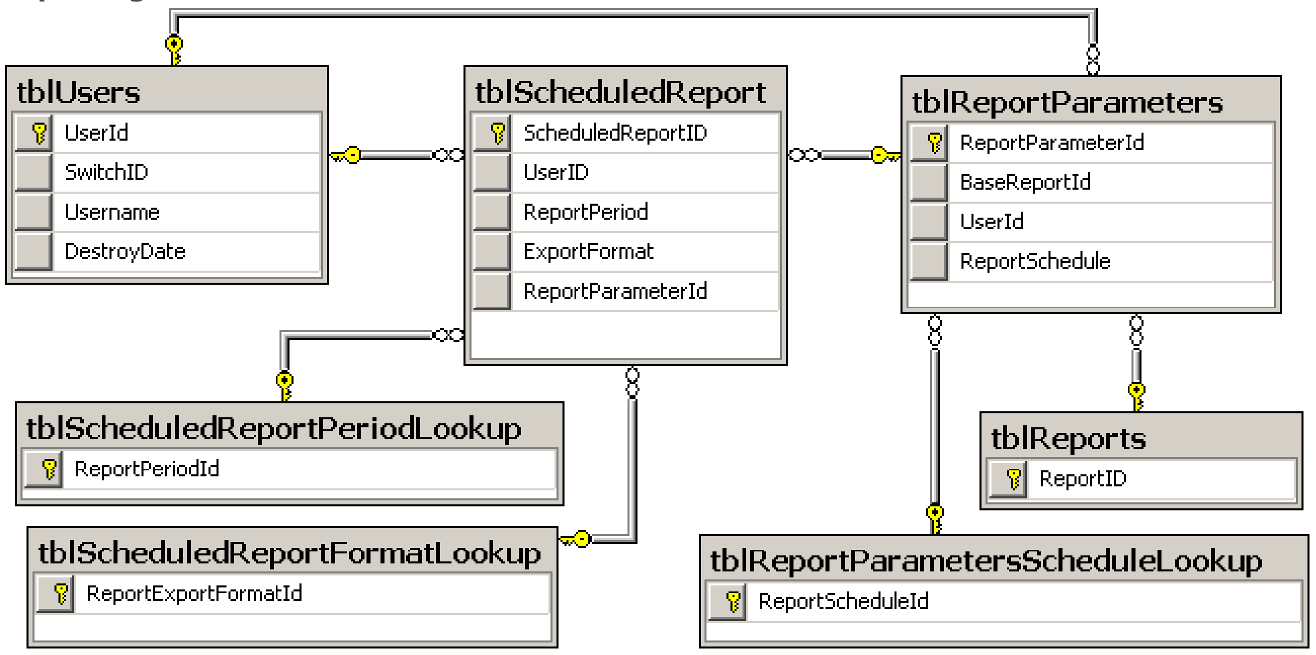
Not all tables are shown in the following diagrams, only those that are used for reports. For a full database diagram, use the Management Studio 8 tool. For the purpose of clarity, the diagram is split in two parts. The first part is about call activity and the second is about the reporting. The common table for both is the User table.

Call Activity



IP Office Customer Call Reporter Database Call Data Section

Reporting



IP Office Customer Call Reporter Database Reporting Section

2.1.1 tblAgentActivity

Column	Data Type	Length	Nullable	Identity	Remarks
AgentActivityID	bigint	8	No	Yes (1,1)	Primary Key for the table.
AgentID	int	4	No	No	Identification of Agent. Get name from tblUsers 21 for AgentID = UserID. Foreign key to tblUsers (Disabled)
HGID	int	4	No	No	Get Hunt Group / Queue description from tblHuntgroup 18 . Foreign key to tblHuntGroup (Disabled)
ReasonCode	nvarchar	8000	Yes	No	Applicable only for ActivityID = 4 23 (Busy not Available). Values are available and configurable using IP Office Manager (System | CCR).
ReasonDescription	nvarchar	8000	Yes	No	Description of Reason Code as configured in IP Office Manager (System | CCR).
ActivityID	smallint	2	No	No	See Activity ID 23 lookup.
StartDate	datetime	8	No	No	Initiation Timestamp of the Activity.
EndDate	datetime	8	Yes	No	Completion Timestamp of the Activity.
CallListID	bigint	8	No	No	"-1" for non call activity (e.g. log in / log out, BNA, etc) and Unique CallListID for call related activities. tblCallList can be JOINed based on CallListID to get further details about the call. Foreign key to tblCallList (Disabled)
IsForced	bit	1	No	No	Not currently used.
SupervisorID	int	4	No	No	Not currently used.
IsModified	bit	1	No	No	Not currently used.
NumberDialed	nvarchar	8000	Yes	No	If the activity is call related and the user dialed a number, this field will be populated.
CallTargetIndex	smallint	2	Yes	No	This is the index of the agent to which call is targeted. This index can change after an event. Example: Huntgroup has 2 agents: Agent1 and Agent2. When the call is presented to the first agent, **CallTargetIndex** will be 1. If the call is refused by Agent1 and presented to agent2, then **CallTargetIndex** will be shown as 2.
CallInformationAction	smallint	2	Yes	No	This bit shows the reason for picking the call by an agent. For example, if it is call pickup or connected due to unheld or unpark. Only following bits are valid. The rest of the bits are not useful for statistics calculations. • Connected = 1 • ConnectedDueToPickUp = 2 • ConnectedDueToUnpark = 3 • ConnectedDueToUnHeld = 4 • ConnectedPostTransfer = 5 • Dialled = 14

2.1.2 tblAgentHGBridge

Column	Data Type	Length	Nullable	Identity	Remarks
HGID	int	4	No	No	Primary Key for the table. Hunt group ID. Foreign key to tblHuntGroup 18
AgentID	int	4	No	No	Primary Key for the table. Agent ID / UserID. Foreign key to tblUsers 21
CreateDate	datetime	8	No	No	Primary Key for the table. Timestamp when Agent became the member of the Hunt group.
DestroyDate	datetime	8	Yes	No	Timestamp when Agent's membership with the hunt group was cancelled.
IsModified	bit	1	No	No	Not currently used.

2.1.3 tblCallList

Column	Data Type	Length	Nullable	Identity	Remarks
CallListID	bigint	8	No	Yes (1,1)	Primary Key for the table. Unique ID for a Call.
CategoryID	smallint	2	No	No	Determines the direction of call. Reference table tblCategoryLookUp 23. Foreign key to tblCategoryLookup
SwitchID	int	4	No	No	Stores the Switch / IPOffice ID. Foreign key to tblSwitch 21.
CampaignID	int	4	No	No	Intended for future use especially with Outbound campaigns.
NumberDialled	nvarchar	8000	Yes	No	Set to Dialed number. This is the number dialed by user where as DDI is the equivalent number assigned by the switch e.g. 8035001 is the number dialed once IPO determines the short code and sends it over SIP like DDI becomes 5001@xxx.xxx.xxx.xxx. That said it is likely to be NULL for inbound calls.
DDI	nvarchar	8000	Yes	No	Dialed number.
CLI	nvarchar	8000	Yes	No	Calling number.
CallerName	nvarchar	8000	Yes	No	Initiating Agent Name
ConnectedID	nvarchar	8000	No	No	Not currently used.
CallID	int	4	No	No	CallID for switch. It is displayed as Reference number in the Call Details report. This ID resets to 1 in certain reboot scenarios of IP Office.
DigitsToCO	nvarchar	8000	Yes	No	Not currently used.
IsCallRecorded	bit	1	No	No	Intended for future use especially with Call Recording.
CreateDate	datetime	8	No	No	If a queue call is not answered by auto-attendant, then the timestamp provides the initiating time which should be referred for calculating Average Abandon time or Average Speed to Answer time.
DestroyDate	datetime	8	Yes	No	Call destruction time.
IsBroken	bit	1	No	No	If a call is cleared in a held state, the isBroken flag is set on the call list. This indicates that a caller hung up while being held. It is difficult to verify the accuracy of this field.
CallbackRequested	bit	1	No	No	Not currently used.
CallCharge	decimal	17	No	No	Not currently used.
IsModified	bit	1	No	No	Not currently used.
IsTransferSetup	bit	1	No	No	Is set for Enquiry Call.
TransferedCallListID	bigint	8	No	No	If this is a transfer setup call, it would specify the call unique identifier [calllistUid] of the call it is trying to transfer.

2.1.4 tblCallEnd

Column	Data Type	Len	Null able	Identity	Remarks
CallEndID	bigint	8	No	Yes (1,1)	Pimary Key to tblCallEnd
CallListID	bigint	8	No	No	ForeignKey for tblCalllist, Unique ID to identify a Call. Foreign key to tblCallList
SwitchID	int	4	No	No	Stores the Switch / IPOffice ID. Foreign key to tblSwitch
IEndFlag	bit	1	No	No	The IEndFlag stands for Initiating End. In the case of an incoming call, the trunk will be the initiating end and will be on the A side. In the case of an outgoing call the Agent is the initiating party and the trunk is the receiving party.
CreateDate	datetime	8	No	No	Time stamp when this call end was created.
DestroyDate	datetime	8	Yes	No	It is the timestamp for the destroyDate of a state. For clearing state, DestroyDate would be always set. For connected state, it will be null.
IsVoicemail	bit	1	No	No	Set when a call is directed from auto attendant(along with IsAnswered) or routed to voicemail (along with IsAnswered and IsVMAnswered).
IsOverflowed	bit	1	No	No	Once the call overflows, this flag is set and will remain set. For overflow lost, IsLost is set and for overflow answered, IsAnswered is set.
OverflowedFromHGID	int	4	No	No	HuntGroupID from which the call overflows.
IsTwinned	bit	1	No	No	Not currently used.
IsManualTransfer	bit	1	No	No	It is set when call is transferred (both supervised and unsupervised). It is set for connected and clearing state in case call is answered. For Lost and Transferred calls, it is set in clearing state. This flag is set for HGID or AgentID as recipient of transfer call. TransferFromHGID and TransferFromAgentID can be used to obtain HGID and agentID who transferred the call.
IsAutoTransfer	bit	1	No	No	It is set for Unsupervised transfer but not used by reporting
IsManualForward	bit	1	No	No	Not currently used.
IsAutoForward	bit	1	No	No	Not currently used.
IsAnswered	bit	1	No	No	Set whenever an end answers a call. It is set when a call is overflowed answered (along with IsOveflowed), when a call is routed to voicemail (along with IsVoicemail and IsVMAnswered), and when a call is directed to auto-attendant (along with IsVoicemail).
IsRefused	bit	1	No	No	Set when call is refused by Agent. When a call is not being answered by an agent within the "No answer time" then this flag get set.
IsMissed	bit	1	No	No	Set when an agent to agent call is lost. It will also be set when an outgoing call is terminated by an agent without being answered by the OutBound End.
IsLost	bit	1	No	No	Set when a call is lost.
IsHGCall	bit	1	No	No	Set for Queue calls.
ConferenceTableID	int	4	No	No	Not currently used. Foreign key to tblConference (Disabled)
VMChannelID	smallint	2	Yes	No	Indicates the voicemail device connected to the call. (You can get available VM channels from tblVoicemailGroup). Foreign key to tblVoicemailChannel (Disabled).
TrunkChannelID	smallint	2	Yes	No	Indicates the trunk device connected to the call. (You can get available trunk channels from tblTrunkGroup). Foreign key to tblTrunkChannel. (Disabled)
HGID	int	4	Yes	No	Set to HuntGroupID. Should not be 0 for Queue calls. JOIN tblHuntGroup to get Huntgroup details.
AgentID	int	4	Yes	No	Set to AgentID. JOIN tblUsers on AgentID = UserID to get Agent details. Foreign key to tblUsers (Disabled)
ExtensionID	int	4	Yes	No	Not currently used.
AccountCode	nvarchar	8000	Yes	No	Column used for GroupBy in reports and target as AccountCode.
IsModified	bit	1	No	No	Not currently used.
StateId	smallint	2	Yes	No	See State ID lookup.

Column	Data Type	Len	Null able	Identity	Remarks
CallEndWaterMark	int	4	Yes	No	Contains internal information for IP Office Customer Call Reporter.
ParkSlot	nvarchar	8000	Yes	No	Park Slot where call is parked.
StateCreateDate	datetime	8	Yes	No	TimeStamp for the corresponding stateID.
VoicemailAnnotation	nvarchar	8000	Yes	No	Stores IVR annotation information. Refers to label from VM module.
Tag	nvarchar	8000	Yes	No	Reserved for future releases.
IsVMAnswered	bit	1	No	No	Set when call is routed to VoiceMail. AgentID should be zero when IsAnswered, IsVoiceMail or IsVMAnswered is set.
IsVMLost	bit	1	No	No	Set when call is lost at VoiceMail.
IsAnsweredOther	bit	1	No	No	Is set when call is answered via call pickup etc. IsAnswered is also set.
FirstAnswered	bit	1	No	No	Set when IsAnswered is set for the first time
FirstTransfer	bit	1	No	No	Indicates that the IsManualTransfer has been set for the first time
OriginalHGID	int	4	Yes	No	First HGID set. Used to identify which was the originalHGID from which the call overflowed. Used when call overflows multiple number of times.
TransferFromAgentID	int	4	Yes	No	AgentID who initiated the transfer for call.
TransferFromHGID	int	4	Yes	No	HGID which initiated the transfer for the call.
VoicemailAgentID	int	4	Yes	No	Set if the call is routed to voicemail by an agent. If Voicemail is on for a user in IP Office Manager configuration, VoicemailAgentID is set to 0 for Queue calls routing to voicemail.
FirstOverflow	bit	1	No	No	Set when call is overflowed for the first time.
IsOverflowing	bit	1	No	No	Set when the call is overflowing. For this record, none of the other flag should be set. For next record, HGID must be set to the Queue to which the call overflows.
OverflowingToHGID	int	4	Yes	No	It is updated with the HuntGroupID to where the call overflows.
TransferToNumber	nvarchar	8000	Yes	No	Set to number to which the call is transferred.
IsRoutingToVoicemail	bit	1	No	No	Set when an end changes from agent or hunt group, it just indicates that the next end will have a voicemail id.
IsTrunkToTrunk	bit	1	No	No	It is set when call from Trunk to Agent (or queue) is transferred to a trunk.
QueueStartTime	datetime	8	Yes	No	The time when this call end entered a queue. The accuracy of this field cannot be verified as it is not used.
FrontEndedByVoicemail	bit	1	No	No	Set when a call is received at auto-attendant first and then routed to Queue or Agent (as per call scenario). It is used to get initiating event for calculation of Average speed to answer or Average Abandon time.
TransferReturn	bit	1	No	No	Set when call is answered after the transfer return that is set in IP Office manager expires. It is only set for connected state.
TransferReturnHGID	int	4	Yes	No	The transfer return hunt group identifier indicates the hunt group where the transfer return has come from.
OverflowIndex	int	4	Yes	No	When a call is marked as overflowing, an index will be placed against the call. When the call is answered, lost or routed to voicemail, the index provided at the fist overflowing point will be provided.
TransferIndex	int	4	Yes	No	When a call is put on hold, an index will be put against the call. When the call is answered, lost or routed to voicemail, the index provided at hold time will be provided.

2.1.5 tblHuntGroup

Column	Data Type	Len	Nullable	Identity	Remarks
HGID	int	4	No	Yes (1,1)	Primary Key for the table. Unique ID for a Hunt Group / Queue.
SwitchID	int	4	No	No	IP Office ID. Foreign key to tblSwitch 21.
Name	nvarchar	8000	No	No	Hunt group / Queue description.
Extension	nvarchar	8000	No	No	Hunt group extension number. Populated from IP Office Manager.
CreateDate	datetime	8	No	No	Timestamp when Hunt group is created.
DestroyDate	datetime	8	Yes	No	Timestamp when a Hunt group is removed.
IsModified	bit	1	No	No	Not currently used.

2.1.6 tblReportParameters

Column	Data Type	Length	Nullable	Identity	Remarks
ReportParameterId	int	4	No	Yes (1,1)	Primary Key for the table.
BaseReportId	smallint	2	No	No	Refers to the basic reports. Foreign key to tblReports 19.
SavedReportName	nvarchar	8000	Yes	No	User defined name for the saved report.
LastRunDate	datetime	8	Yes	No	Timestamp when last executed.
NextRunDate	datetime	8	Yes	No	Timestamp for next scheduled execution.
LastModifiedDate	datetime	8	No	No	Timestamp when last update made.
StartDate	datetime	8	No	No	Report Period Start Date.
EndDate	datetime	8	No	No	Report Period End Date.
StartTime	nvarchar	8000	No	No	Report Period Start Time.
EndTime	nvarchar	8000	No	No	Report Period End Time.
TargetId	int	4	No	No	See Target ID lookup.
GroupId	int	4	No	No	See Group ID lookup.
FilterId	int	4	No	No	See Filter ID lookup.
UserId	int	4	No	No	User who scheduled the report. Foreign key to tblUsers 21.
ReportSchedule	smallint	2	No	No	Stores information about how the report is scheduled. Foreign key to tblReportParametersScheduleLookup 18.
IncludeInternal	bit	1	No	No	Flag to indicate internal call.
IncludeSaturdays	bit	1	No	No	Flag to indicate Saturday.
IncludeSundays	bit	1	No	No	Flag to indicate Sunday.
TargetValue	nvarchar	8000	No	No	Target value specified for the report.
ReportLanguage	nchar	8000	Yes	No	Report language option selected.
GraphReportOptions	nchar	8000	Yes	No	Not currently used.
ASAThreshold	int	4	Yes	No	Average Answer Time Threshold applicable for Call Summary Report only. Not currently used.
LostCallThreshold	int	4	Yes	No	Lost Call Threshold applicable for Call Summary Report only. Not currently used.
MinTalkTreshold	int	4	Yes	No	Threshold for APF calculations. Not currently used.
MaxTalkTreshold	int	4	Yes	No	Threshold for APF calculations. Not currently used.
CustomReportName	nvarchar	8000	Yes	No	Name given to a custom report.
CCRVersion	nvarchar	8000	No	No	Last CCR version in which a database schema change was made.

2.1.7 tblReportParametersScheduleLookup

Column	Data Type	Length	Nullable	Identity	Remarks
ReportScheduleId	smallint	2	No	No	Primary Key for the table.
ReportScheduleName	varchar	8000	No	No	Schedule description.

2.1.8 tblReports

Column	Data Type	Len	Nullable	Identity	Remarks
ReportID	smallint	2	No	No	Primary Key, referenced by BaseReportId of tblReportParameters 18.
ReportTitle	nvarchar	8000	No	No	Resource key for report name, as rendered in web client, typically prefixed by DB5_.
ReportKey	char	8000	No	No	Not currently used.
ReportTemplateName	nvarchar	8000	No	No	Name of the Crystal Report .rpt file

2.1.9 tblScheduledReport

Column	Data Type	Length	Nullable	Identity	Remarks
ScheduledReportID	int	4	No	Yes (1,1)	Primary Key for the table.
UserID	int	4	No	No	UserID who scheduled the report. Foreign key to tblUsers 21
Frequency	smallint	2	Yes	No	1=Daily, 2=Weekly, 3=Monthly, 4=Unscheduled (currently no database lookup table).
ReportPeriod	smallint	2	No	No	Relates how the report will be scheduled like Daily, Weekly etc. Refer tblScheduledReportPeriodLookup 19. Foreign key to tblScheduledReportPeriodLookup
ReportPeriodCount	smallint	2	No	No	Report content set during report saving.
StartTime	nvarchar	8000	Yes	No	Time when the task will be started.
PrinterName	nvarchar	8000	Yes	No	Name of the Printer where the report will be printed.
EmailList	nvarchar	8000	Yes	No	Email ID where exported report will be mailed.
ExportFormat	smallint	2	No	No	Format to export the report. Refer tblScheduledReportFormatLookup 19. Foreign key to tblScheduledReportFormatLookup
PrintNoOfCopies	smallint	2	Yes	No	Nunber of copies of report.
WeeklyDayOfWeek	smallint	2	No	No	If scheduled weekly, specific day of week to execute, 0=Sunday to 6=Saturday.
MonthlyOption	smallint	2	No	No	If scheduled monthly, 1=Specific day of month by date, 2=Specific day based on days and weeks in month, e.g. 3rd Tuesday of month.
MonthlyDayOfMonth	smallint	2	No	No	If scheduled monthly, specific day of month to execute, date-1, e.g. 20th = 19.
MonthlyOccurence	smallint	2	No	No	If scheduled monthly, based on days and weeks in month, 0=First, 1=Second, 2=Third, 3=Fourth, 4=Last.
MonthlyDayOfWeek	smallint	2	No	No	If scheduled monthly, based on days and weeks in month, 0=Sunday to 6=Saturday.
ReportParameterId	int	4	No	No	Stores the parameters saved for the report. Foreign key to tblReportParameters.
DailyIncludesWeekend	bit	1	No	No	Includes weekend or not for daily reports.
IncludeCurrentDay	bit	1	No	No	Includes current day in scheduled reports.

2.1.10 tblScheduledReportPeriodLookup

Column	Data Type	Length	Nullable	Identity	Remarks
ReportPeriodId	smallint	2	No	No	Primary Key for the table. Referenced in tblScheduledReport 19.
ReportPeriodName	varchar	8000	No	No	Name of Report Period e.g. Daily, Weekly, etc.

2.1.11 tblScheduledReportFormatLookup

Column	Data Type	Length	Nullable	Identity	Remarks
ReportExportFormatId	smallint	2	No	No	Primary Key for the table. Referenced in tblScheduledReport 19.

Column	Data Type	Length	Nullable	Identity	Remarks
ReportExportFormatName	varchar	8000	No	No	List of the possible report export format.

2.1.12 tblSwitch

Column	Data Type	Length	Nullable	Identity	Remarks
SwitchID	int	4	No	Yes (1,1)	Primary Key for the table. Unique ID for the switch (IP Office).
FirmwareVersion	nvarchar	8000	Yes	No	IP Office Core version.
IP	nvarchar	8000	No	No	IP address of the IP Office.
Name	nvarchar	8000	Yes	No	Name of the IP Office.
LastConfigMerge	datetime	8	Yes	No	Indicates last configuration merge time. The accuracy cannot be verified.
CreateDate	datetime	8	No	No	Timestamp when the IP Office connected to the IP Office Customer Call Reporter application.
DestroyDate	datetime	8	Yes	No	Timestamp when the IP Office disconnected from IP Office Customer Call Reporter application.
MacAddressUpper	bigint	8	No	No	Specifies upper half of Switch MAC address.
MacAddressLower	bigint	8	No	No	Specifies lower half of Switch MAC address.
DAIP	nvarchar	8000	Yes	No	IP address of the server where DA (Data Analyzer) Service is running. This should be the IP address of the server where IP Office Customer Call Reporter application is installed.
SSIUserName	nvarchar	8000	No	No	Username for IP Office.
SSIPassword	nvarchar	8000	Yes	No	Password for IP Office.
DomainName	nvarchar	8000	Yes	No	Domain name of the Switch.
IsModified	bit	1	No	No	Not currently used.
StatusID	int	4	No	No	Required for IP Office Customer Call Reporter internal purpose.
LocationLatitude	float	8	No	No	Not used.
LocationLongitude	float	8	No	No	No used.
LocationPublic	bit	1	No	No	Not used.
LocationDescription	nvarchar	8000	Yes	No	Not used.

2.1.13 tblUsers

Column	Data Type	Length	Nullable	Identity	Remarks
UserId	int	4	No	Yes (1,1)	Primary Key for the table. Unique ID for an Agent / Supervisor / Wallboard User / Admin.
SwitchID	int	4	Yes	No	IP Office ID. Foreign key to tblSwitch 21.
Username	nvarchar	8000	No	No	Name of the User (Agent, Supervisor, etc).
Password	nvarchar	8000	Yes	No	Encrypted Password for users to login into IP Office Customer Call Reporter, mainly for Supervisors, administrators, etc.
Roles	int	4	No	No	To distinguish Agent / Supervisor/ Admin etc.
FullName	nvarchar	8000	No	No	Full name of the user.
EmailID	nvarchar	8000	No	No	Mail ID for User.
Extension	nvarchar	8000	No	No	Agents extension.
CreateDate	datetime	8	No	No	Timestamp when User is created.
DestroyDate	datetime	8	Yes	No	Timestamp when User is removed.
SelfAdministrateViews	bit	1	No	No	Setting info for User Account attribute / status.
Enabled	bit	1	No	No	Setting info for User Account attribute / status.
ResetStatistics	bit	1	No	No	Setting info for User Account attribute / status.
Display	bit	1	No	No	Setting info for User Account attribute / status.

Column	Data Type	Length	Nullable	Identity	Remarks
Audio	bit	1	Yes	No	Setting info for User Account attribute / status.
HelpTooltips	bit	1	Yes	No	Setting info for User Account attribute / status.
HighlightStatistics	bit	1	No	No	Setting info for User Account attribute / status.
ForceAgentState	bit	1	No	No	This field enables set state dialog in the real time for controlling the agent state.
RecentReportsArchiveDays	smallint	2	No	No	Setting info for User Account attribute / status
OpenReportsInNewWindow	bit	1	No	No	Setting info for User Account attribute / status
ShowLoggedOffAgents	bit	1	No	No	Setting info for User Account attribute / status. Not currently used.

2.1.14 Lookup Tables

Lookup tables are used to provide a mapping between human readable values and values stored in other tables. This allows the other tables to store simple numerical values rather than long strings. The meaning of the numeric value is determined by reference to the appropriate lookup table.

2.1.14.1 tblActivityLookup

Activity ID	Activity Description
1	Idle / Ready
2	Ringing / Alerting
3	Incoming
4	Busy Not Available
5	Hold
6	ACW
7	Logged Off
8	Logged In
9	Busy
10	Outgoing
11	Internal Made
12	Internal Received
13	Enable in hunt group
14	Disabled in hunt group

2.1.14.2 tblCategoryLookup

CategoryID	Description
1	Outgoing
2	Incoming
3	Internal

2.1.14.3 tblReportFilters

Report	FilterId	Description
Agent Time Card	1	All
	9	Shifts
	10	Lunch
	11	Breaks
	12	Talk Time
	13	Performance
	14	Calls
Call Details Report	1	All
	2	Answered
	3	No answer
	4	Overflowed Lost
	5	Overflowed Answered
	6	Transferred
	7	Lost Calls
	8	Routed to voicemail
	15	New Calls
	16	Holding
	17	Enquiry Answered
	18	Not Answered
	19	Connected

Call Summary Report	1	All

2.1.14.4 tblReportGroups

Report	GroupId	Description
Agent Summary Report	7	QUEUE
Agent Time Card	5	DAY
	6	WEEK
	7	AGENT
Call Details Report	1	UNGROUPED
	2	15 MINUTES
	3	30 MINUTES
	4	HOUR
	5	DAY
	6	WEEK
	7	QUEUE
	8	AGENT
	9	CLI
	10	DDI
	11	ACCOUNT CODE
Call Summary Report	1	UNGROUPED
	2	15 MINUTES
	3	30 MINUTES
	4	HOUR
	5	DAY
	6	WEEK
	7	QUEUE
	8	AGENT
	9	CLI
	10	DDI
	11	ACCOUNT CODE
Voicemail Report	1	UNGROUPED
	4	HOUR
	5	DAY
	6	WEEK
	9	CLI
	10	DDI

2.1.14.5 tblScheduledReportPeriodLookup

StateID	Description
0	Daily
1	Weekly
2	Monthly

2.1.14.6 tblScheduledReportFormatLookup

ReportExportFormatID	Description
0	PDF
1	MS Word (Read only)
2	MS Excel (Data only)
3	Rich Text Format
4	Crystal
5	MS Word (Editable)
6	MS Excel (Data only)

7	XML
8	CSV
9	HTML
10	Text

2.1.14.7 tblReportTargets

Report	TargetId	Description
Agent Summary Report	1	Queue
	2	View
	3	Agent
Agent Time Card	3	Agent
Call Details Report	1	Queue
	2	View
	3	Agent
	4	DDI
	5	CLI
	6	Account code
Call Summary Report	1	Queue
	2	View
	3	Agent
	4	DDI
	5	CLI
	6	Account code
Trace Report	3	Agent
	5	CLI
	9	Call reference
Voicemail Report	8	Voicemail

2.1.14.8 tblStateLookup

StateID	Description
2	Connected
3	Hold
9	Seized
10	Dialing
16	Ringing
18	Queuing
19	Clearing

2.2 Stored Procedures

There are numerous Stored Procedures associated with the IP Office Customer Call Reporter database. Those can be used by the application written to create Custom Reports. Note that any modifications to the these will break IP Office Customer Call Reporter functionality. These should only be used as references if new stored procedures need to be created for the custom report.

The following is a list of the stored procedures used by IP Office Customer Call Reporter. The parameters for those functions can be seen using Management Studio 9.

S. No.	Procedure	S. No.	Procedure
1	spAddAgent	**120**	spManagementServiceCreateWallboard
2	spAddAgentActivity	**121**	spManagementServiceCreateWallboardUser
3	spAddAgentsToHG	**122**	spManagementServiceDeleteHuntGroupSupervisorBridge
4	spAddAlarm	**123**	spManagementServiceDeleteHuntGroupView
5	spAddAlarmDetails	**124**	spManagementServiceDeleteSignOn
6	spAddCallEnd	**125**	spManagementServiceDeleteSwitch
7	spAddCallList	**126**	spManagementServiceDeleteTrunkGroupSupervisorBridge
8	spAddConference	**127**	spManagementServiceDestroyStatLookupViewBridge
9	spAddExtension	**128**	spManagementServiceDestroySupervisor
10	spAddExtensionsToHG	**129**	spManagementServiceDestroyUser
11	spAddHG	**130**	spManagementServiceDestroyWallboardUser
12	spAddRTRequest	**131**	spManagementServiceGetActivities
13	spAddRTStat	**132**	spManagementServiceGetAllCSRs
14	spAddSwitch	**133**	spManagementServiceGetAllSignOnTimeOut
15	spAddSwitchActivity	**134**	spManagementServiceGetAllSignOnTimeOutRole
16	spAddSwitchWithPendingStatus	**135**	spManagementServiceGetAllSupervisors
17	spAddTrunkChannel	**136**	spManagementServiceGetAllSystemSettings
18	spAddTrunkGroup	**137**	spManagementServiceGetAllWallboards
19	spAddTrunkGroupBusyTime	**138**	spManagementServiceGetAllWallboardsUnlocked
20	spAddVMChannel	**139**	spManagementServiceGetAllWallboardUsers
21	spAddVMGroup	**140**	spManagementServiceGetCSR
22	spAddVMSelection	**141**	spManagementServiceGetCSRsInHuntGroup
23	spAgentSummaryReport	**142**	spManagementServiceGetDashboardGoal
24	spAgentSummaryReportHGEnabled	**143**	spManagementServiceGetDashboardPanes
25	spAgentSummaryReportHGTotals	**144**	spManagementServiceGetDatabaseVersion
26	spAgentSummaryReportNonHGTotals	**145**	spManagementServiceGetHuntGroups
27	spAgentTimeCardReport	**146**	spManagementServiceGetHuntGroupsForCSR
28	spAlarmReport	**147**	spManagementServiceGetHuntGroupsSupervisor
29	spATCRGetAvailabilityDetails	**148**	spManagementServiceGetHuntGroupStates
30	spATCRGetAvailabilityDuration	**149**	spManagementServiceGetHuntGroupsView
31	spATCRGetInboundCallStats	**150**	spManagementServiceGetMonitoringAlarms
32	spCallDetailReport	**151**	spManagementServiceGetPassword
33	spCallDetailReportForAccountCode	**152**	spManagementServiceGetSchemaVersion
34	spCallDetailReportForAgentORCSR	**153**	spManagementServiceGetSignOnTimeOut
35	spCallDetailReportForCLIDDI	**154**	spManagementServiceGetSignOnTimeOutPerSession
36	spCallDetailReportForHuntGroup	**155**	spManagementServiceGetSignOnTimeOutPerUserSession
37	spCallDetailReportForView	**156**	spManagementServiceGetStatLookup
38	spCallSummaryReport	**157**	spManagementServiceGetStatLookupView
39	spCallSummaryReportForAccountCode	**158**	spManagementServiceGetStatParameters
40	spCallSummaryReportForCLI	**159**	spManagementServiceGetSuperAdmin
41	spCallSummaryReportForCSR	**160**	spManagementServiceGetSupervisor
42	spCallSummaryReportForDDI	**161**	spManagementServiceGetSupervisorViews
43	spCallSummaryReportForHuntGroup	**162**	spManagementServiceGetSupervisorViewSettings
44	spCallSummaryReportForView	**163**	spManagementServiceGetSystemSetting
45	spClearCache	**164**	spManagementServiceGetTimeOut
46	spClearConference	**165**	spManagementServiceGetTrunkGroups
47	spCustomReportsCallAnswerDuration	**166**	spManagementServiceGetTrunkGroupsSupervisor
48	spCustomReportsCallBasic	**167**	spManagementServiceGetUserBase
49	spCustomReportsCallDuration	**168**	spManagementServiceGetUserId
50	spCustomReportsCallHeldTime	**169**	spManagementServiceGetUserRoles
51	spCustomReportsCallOverflow	**170**	spManagementServiceGetViewSettingsHGStateThreshold

S. No.	Procedure
52	spCustomReportsCallQueueTime
53	spCustomReportsCallStatus
54	spCustomReportsCallTransfer
55	spCustomReportsQueueBasic
56	spCustomReportsQueueThresholdDependent
57	spCustomReportsQueueThresholdDependentIntervals
58	spCustomReportsQueueTimes
59	spCustomReportsSystemBasic
60	spCustomReportsSystemOutgoing
61	spCustomReportsSystemThresholdDependent
62	spCustomReportsSystemThresholdDependentIntervals
63	spCustomReportsSystemTimes
64	spCustomReportsVoicemail
65	spDatabaseMonitorDeleteAll
66	spDatabaseMonitorDeleteCallData
67	spDatabaseMonitorDeleteHuntTrunkGroupsSwitches
68	spDatabaseMonitorDeleteOldestPercentageCallData
69	spDatabaseMonitorDeleteSafe
70	spDatabaseMonitorDeleteUsersViewsReports
71	spDatabaseMonitorGetSize
72	spDatabaseMonitorMain
73	spDatabaseMonitorRebuildIndexes
74	spDeleteAlarm
75	spDeleteAlarmById
76	spDeleteMaintenance
77	spDeleteSavedReport
78	spEndAgentActivity
79	spGetActiveAgentHG
80	spGetActiveAgents
81	spGetActiveExtensionHG
82	spGetActiveExtensions
83	spGetActiveHGs
84	spGetActiveSwitches
85	spGetActiveTrunks
86	spGetActiveVMs
87	spGetAlarms
88	spGetAllLastRunTasks
89	spGetAllSupervisors
90	spGetAllSupervisorViews
91	spGetAllSwitches
92	spGetAllTargets
93	spGetCategoryLookup
94	spGetCLIs
95	spGetLastRunReports
96	spGetMaintenanceProperties
97	spGetMaintenanceTasks
98	spGetProcessId
99	spGetReportInfo
100	spGetReportParameters
101	spGetSavedReportParameters
102	spGetScheduledReports
103	spGetScheduleProperties
104	spGetStatGroupLookup
105	spGetStatLookup
106	spGetSubjects
107	spGetTargetFilterInAndLikeLiterals
108	spGetTargetFilters
109	spGetTargetList
171	spManagementServiceGetViewSettingsStateThreshold
172	spManagementServiceGetWallboard
173	spManagementServiceGetWallboardByUniqueID
174	spManagementServiceGetWallboardLock
175	spManagementServiceGetWallboardsSupervisor
176	spManagementServiceListAllSwitches
177	spManagementServiceResetPassword
178	spManagementServiceResetPasswordByUsername
179	spManagementServiceRollSignOn
180	spManagementServiceSetDashboardGoal
181	spManagementServiceSetDashboardPane
182	spManagementServiceSetHuntGroupSupervisorBridge
183	spManagementServiceSetHuntgroupViewBridge
184	spManagementServiceSetStatLookupViewBridge
185	spManagementServiceSetStatParameters
186	spManagementServiceSetSupervisorView
187	spManagementServiceSetSupervisorViewSettings
188	spManagementServiceSetSystemSetting
189	spManagementServiceSetTrunkGroupSupervisorBridge
190	spManagementServiceSetViewSettingsHGStateThreshold
191	spManagementServiceSetViewSettingsStateThreshold
192	spManagementServiceSetWallboardLock
193	spManagementServiceUpdateCSR
194	spManagementServiceUpdateSuperAdmin
195	spManagementServiceUpdateSupervisor
196	spManagementServiceUpdateWallboard
197	spManagementServiceVerifySuperAdmin
198	spManagementServiceVerifyWallboardUser
199	spRemoveAgent
200	spRemoveAgentsFromHG
201	spRemoveExtension
202	spRemoveExtensionsFromHG
203	spRemoveHG
204	spRemovePendingSwitch
205	spRemoveRTRequest
206	spRemoveSwitch
207	spRemoveTrunkGroup
208	spRemoveVMGroup
209	spSaveLastRunReport
210	spSaveReportParameters
211	spSaveScheduleProperties
212	spSwitchDisconnectivityDetail
213	spSystemGetAnsweredCall
214	spSystemGetLostCall
215	spSystemUpdateAnsweredCall
216	spSystemUpdateLostCall
217	spTraceReport
218	spUpdateAgentActivity
219	spUpdateAlarmDetails
220	spUpdateAlarmStatus
221	spUpdateAlarmThresholds
222	spUpdateCallList
223	spUpdateMaintenanceProperties
224	spUpdateReportParameters
225	spUpdateScheduleProperties
226	spUpdateStatValue
227	spUpdateSwitchConnection
228	spUpdateSwitchConnectionStatus

S. No.	Procedure
110	spGetViewThresholdsForHGCollection
111	spGraphReport
112	spInitializeSPInput
113	spKillProcessId
114	spLastStatsReset
115	spListReports
116	spManagementServiceChangePassword
117	spManagementServiceCreateSignOn
118	spManagementServiceCreateSuperAdmin
119	spManagementServiceCreateSupervisor

S. No.	Procedure
229	spUpdateSwitchDetails
230	spUpdateSwitchLocation
231	spUpdateSwitchParameters
232	spUpdateSwitchWithPendingStatus
233	spVoiceMailReport
234	spWallBoardMessageAddMessage
235	spWallBoardMessageDeleteMessage
236	spWallBoardMessageGetCurrentMessages
237	spWallBoardMessageGetMessage
238	spWallBoardMessageUpdateMessage

2.3 User Defined Functions

There are numerous Functions associated with the IP Office Customer Call Reporter database. Those can be used by the custom application to create custom reports. Note that any modifications to the existing Functions will break IP Office Customer Call Reporter functionality. Existing functions should only be used as reference examples if new functions need to be created for the custom report.

Here is the list of the functions used by IP Office Customer Call Reporter (table valued and scalar valued). The source for those functions can be seen using Management Studio 10.

S. No.	Table Functions
1	Split
2	udfATCGetRefusedCountperAgent
3	udfATCRGetAgentData
4	udfATCRGetAvailabilityDetails
5	udfATCRGetAvailabilityDuration
6	udfATCRGetCallStats
7	udfATCRGetCallsWithinTalkThreshold
8	udfATCRGetInboundCallStats
9	udfATCRGetOutboundCallStats
10	udfATCRGetTalkDuration
11	udfATCRGetTransferSetupCallStats
12	udfCDRGetAllCallDuration
13	udfCDRGetAllCalls
14	udfCDRGetAllCallsForAgentOrCSR
15	udfCDRGetAllCallsForHuntGroupView
16	udfCDRGetAnsweredCallAfterHoldForTransfer
17	udfCDRGetAnsweredCalls
18	udfCDRGetAnsweredDuration
19	udfCDRGetHeldCalls
20	udfCDRGetHeldDuration
21	udfCDRGetLostCalls
22	udfCDRGetMainCallView
23	udfCDRGetMainCallViewForAccountCode
24	udfCDRGetMainCallViewForAgentORCSR
25	udfCDRGetMainCallViewForCLIDDI
26	udfCDRGetMainCallViewForHuntGroups
27	udfCDRGetMainCallViewForOverFlowedHuntGroups
28	udfCDRGetMainCallViewForOverFlowedHuntGroups1
29	udfCDRGetMainCallViewForOverFlowedViews
30	udfCDRGetMainCallViewForViews
31	udfCDRGetNewCallsForAccountCode
32	udfCDRGetNewCallsForAgentORCSR
33	udfCDRGetNewCallsForCLIDDI
34	udfCDRGetNewCallsForHuntGroups
35	udfCDRGetNewCallsForView
36	udfCDRGetOverflowedAnsweredCalls
37	udfCDRGetOverflowedLostCalls
38	udfCDRGetOverflowingCalls
39	udfCDRGetOverflowingDetails
40	udfCDRGetQueueTime
41	udfCDRGetRefusedCalls
42	udfCDRGetRoutedToVMCSR
43	udfCDRGetTransferredCalls
44	udfCDRGetTransferredCallsForHuntGroupView
45	udfCDRGetTransferredDetails
46	udfCDRGetVoicemailedCalls
47	udfCSRGetAbandonTime
48	udfCSRGetAbandonTimeForTransfer
49	udfCSRGetAgentAnswerTimeForNonQueueCalls
50	udfCSRGetAgentAnswerTimeForNormalQueueCalls

S. No.	Scalar Functions
1	udf_get_AnsTime
2	udf_get_reportGeneric
3	udf_get_targetValue
4	udfATCRGetActivityDetails
5	udfDupLoginFilter
6	udfGetActivityDuration
7	udfGetLoginDate
8	udfGetLogOffEvent

S. No.	Table Functions
51	udfCSRGetAgentAnswerTimeForTransfer
52	udfCSRGetAnsweredCalls
53	udfCSRGetAnswerTimeForIVRAnswer
54	udfCSRGetAnswerTimeForIVRNoAnswer
55	udfCSRGetAnswerTimeForTransfer
56	udfCSRGetHuntGroupName
57	udfCSRGetLostCalls
58	udfCSRGetMainCallViewForAgentORCSR
59	udfCSRGetMainCallViewForCLIDDIAccountCode
60	udfCSRGetMainCallViewForHuntGroup
61	udfCSRGetMainCallViewForView
62	udfCSRGetOutgoingCalls
63	udfCSRGetOverflowedAnsweredCalls
64	udfCSRGetOverflowedLostCalls
65	udfCSRGetRefusedCalls
66	udfCSRGetTransferReturn
67	udfCSRGetUserName
68	udfCSRGetVoiceMailCalls

Chapter 3. Example

3. Example

3.1 Development Environment

The development of the application that will mine the database can be done using any environment that provides access to the interface required to access the SQL database. If Microsoft is used, here are some useful URLs:

- **Data Development Center:** http://msdn.microsoft.com/en-us/data/default.aspx
- **Data Technologies Overview:** http://msdn.microsoft.com/library/ee730344.aspx
- **ADO.NET:** http://msdn.microsoft.com/en-us/library/aa286484(v=MSDN.10).aspx
- **LINQ to SQL:** http://msdn.microsoft.com/en-us/library/bb386976.aspx

3.2 Data Calculation

The information stored in the database can be used to calculate information that is required in reports.

The following table provides some logic on how to get information from the database.

Item	Description	Implemented Logic
HG Enabled Time	The duration for which an agent is enabled in a Hunt Group.	Difference between StartDate for ActivityID = 13 (Enable In Hunt Group) & Immediate next StartDate for ActivityID = 14 (Disable in Hunt Group) / 7 (Logged Off)
Ringing Time	Ring time of calls directed to the Queue. This is a hunt group specific attribute.	Difference between StartDate & EndDate for ActivityID = 2 (Ringing)
Talk Outbound	External calls only, does not include internal calls. This is a non hunt group specific attribute.	Difference between StartDate & EndDate for ActivityID = 10 (Outgoing) + Difference between StartDate & EndDate for ActivityID = 9 (Busy) for the same call
Talk Inbound	Talk time on calls answered for the queue. This can be a hunt group specific and/or non – hunt group attribute.	Difference between StartDate & EndDate for ActivityID = 12 (Internal Received) + Difference between StartDate & EndDate for ActivityID = 3 (Incoming) for the same call
Talk Internal	Talk time on call made to another internal party. This is a non hunt group specific attribute.	Difference between StartDate & EndDate for ActivityID = 11 (Internal Made)
Busy Not Available	Duration of telephone in Busy State. This is a non hunt group specific attribute.	Difference between StartDate & EndDate for ActivityID = 4 (Busy Not Available)
ACW Time	Duration for After Call Work (ACW). This is a non hunt group specific attribute.	Difference between StartDate & EndDate for ActivityID = 6 (After Call Work)
Hold Time	Holding includes park	If StartDate <> EndDate for ActivityID = 5 (Hold) then: Difference between StartDate & EndDate for ActivityID =5 (Hold) Else Difference between StartDate of ActivityID = 5 & StartDate of very next Activity after ActivityID = 5 for the same call (The next activity is ActivityID = 9 (Busy))
Off Hook Time	Includes picking up handset, dialing and ring time. For a trunk it is the time until the trunk is seized. This is a non hunt group specific attribute.	Difference between StartDate & EndDate for ActivityID = 9 (Busy)
Non Queue Time	Direct inbound call including ring time. This is a non hunt group specific attribute.	Difference between StartDate & EndDate for ActivityID = 12 (Internal Received) + Difference between StartDate & EndDate for ActivityID = 3 (Incoming) + Difference between StartDate & EndDate for ActivityID = 2 (Ringing)

3.3 Sample Code

The following sample code taken from IP Office Customer Call Reporter is used to generate the Agent Summary Report.

3.3.1 Stored Procedure

First, here are the stored procedure parameters and code for spAgentSummaryReport.

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
GO

-- =============================================
-- Description:      Generates the agent summary report
-- =============================================
ALTER PROCEDURE [dbo].[spAgentSummaryReport]
         @Target nvarchar(50), --can be one of the following - CLI,DDI,Hunt
                               --Group,CSR,Account Code,View
         @TargetValue nvarchar(MAX),
         @IncludeSaturday bit, --1 = include , 0 = exclude
         @IncludeSunday bit, --1 = include , 0 = exclude
         @FromDate datetime,
         @ToDate   datetime,
         @StartTime smalldatetime,
         @EndTime smalldatetime,
         @SupervisorId bigint
AS
BEGIN
       SET NOCOUNT ON
       SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
       SET DATEFIRST 7

DECLARE @DStartTime DATETIME,@DEndTime DATETIME, @IsTimeSpanOverMidNight BIT

-- Since these are datetime variables and we are extracting only start time and end time,
-- sql server would append default date to these variables, i.e. Jan  1 1900.
SET @DStartTime = CONVERT(char(5), @FromDate, 8)
SET @DEndTime = CONVERT(char(5), @ToDate, 8)

-- Set the timespan parameters
SET @IsTimeSpanOverMidNight = CASE
                             WHEN @DStartTime < @DEndTime THEN 0
                             ELSE 1
                           END

-- The switch disconnectivity is to be shown on report template.
-- Call the sp spSwitchDisconnectivityDetail to get the details that
-- need to be shown on report.
EXEC spSwitchDisconnectivityDetail @FromDate, @ToDate, @DStartTime, @DEndTime, @IsTimeSpanOverMidNight


       DECLARE @SPID varbinary(128);
       SELECT @SPID = CAST(CAST(@@SPID as varchar(10)) as varbinary(128));
       SET CONTEXT_INFO @SPID;

       SELECT @Target = LTRIM(RTRIM(@Target))
       SELECT @TargetValue = LTRIM(RTRIM(@TargetValue))

       --Create Temporary Table
       CREATE TABLE #agentSummary
       (
              AgentId bigint,
              AgentName varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS,
              HuntgroupId bigint,
              HuntgroupName varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS,
              OtherTime bigint,
              RingTime bigint,
              Outbound bigint,
              Inbound bigint,
              Internal bigint,
              BusyNotAvailableTime bigint,
              ACWTime bigint,
              HoldTime bigint,
              OffHookTime bigint,
              HGEnabled bigint
       );

       --Declare some variables
       --DECLARE @LogInTime bigint
       DECLARE @OtherTime bigint
       DECLARE @RingTime bigint
       DECLARE @Outbound bigint
       DECLARE @Inbound bigint
       DECLARE @Internal bigint
       DECLARE @BusyNotAvailableTime bigint
       DECLARE @ACWTime bigint
       DECLARE @HoldTime bigint
       DECLARE @OffHookTime bigint
       DECLARE @DaysOfWeek varchar(13);
       DECLARE @HGEnabled bigint;

       --Set Days of the week
       SET @DaysOfWeek = '2,3,4,5,6';
       IF(@IncludeSaturday=1)
```

```
   SET @DaysOfWeek = @DaysOfWeek + ',7';

IF(@IncludeSunday=1)
   SET @DaysOfWeek = '1,' + @DaysOfWeek;

--Check For Wildcard
DECLARE @StarPos int;
SET @StarPos = 0;
IF (@TargetValue <> '*')
BEGIN
   SET @StarPos = CHARINDEX('*' , @TargetValue)
   IF @StarPos > 0
   BEGIN
      SET @TargetValue = REPLACE(@TargetValue,'*','%')
   END
END
--Get Agent List
--For Views
IF (@Target = 'View')
BEGIN
   --Wildcard ALL
   IF (@TargetValue = '*')
   BEGIN
      DECLARE cur CURSOR FOR
         SELECT DISTINCT AgentId, Username COLLATE SQL_Latin1_General_CP1_CI_AS,
                         tblHuntgroup.HGID,
                         tblHuntgroup.[Name] COLLATE SQL_Latin1_General_CP1_CI_AS,
                         FullName COLLATE SQL_Latin1_General_CP1_CI_AS
         FROM tblSupervisorView
         JOIN tblHGViewBridge ON tblSupervisorView.ViewId =
                                  tblHGViewBridge.ViewId
         JOIN tblHuntgroup ON tblHGViewBridge.HGID = tblHuntgroup.HGID
         JOIN tblAgentHGBridge ON tblHuntgroup.HGID = tblAgentHGBridge.HGID
         JOIN tblUsers ON tblAgentHGBridge.AgentId = tblUsers.UserId
         WHERE tblSupervisorView.DestroyDate IS NULL
              AND tblHuntgroup.DestroyDate IS NULL
              AND tblUsers.DestroyDate IS NULL
              AND (tblAgentHGBridge.DestroyDate IS NULL OR
                     tblAgentHGBridge.DestroyDate > @FromDate)
              AND tblSupervisorView.SupervisorId = @SupervisorId
         ORDER BY HGID
      FOR READ ONLY;
  END
  --Wildcard with a word
  ELSE IF (@StarPos > 0)
   BEGIN
     DECLARE cur CURSOR FOR
        SELECT DISTINCT AgentId, Username COLLATE SQL_Latin1_General_CP1_CI_AS,
                        tblHuntgroup.HGID,
                        tblHuntgroup.[Name] COLLATE SQL_Latin1_General_CP1_CI_AS,
                        FullName COLLATE SQL_Latin1_General_CP1_CI_AS
        FROM tblSupervisorView
        JOIN tblHGViewBridge ON tblSupervisorView.ViewId = tblHGViewBridge.ViewId
        JOIN tblHuntgroup ON tblHGViewBridge.HGID = tblHuntgroup.HGID
        JOIN tblAgentHGBridge ON tblHuntgroup.HGID = tblAgentHGBridge.HGID
        JOIN tblUsers ON tblAgentHGBridge.AgentId = tblUsers.UserId
        WHERE tblSupervisorView.DestroyDate IS NULL
             AND tblHuntgroup.DestroyDate IS NULL
             AND tblUsers.DestroyDate IS NULL
             AND (tblAgentHGBridge.DestroyDate IS NULL OR
                   tblAgentHGBridge.DestroyDate > @FromDate)
             AND tblSupervisorView.[Name] COLLATE SQL_Latin1_General_CP1_CI_AS
                  LIKE @TargetValue
             AND tblSupervisorView.SupervisorId = @SupervisorId
        ORDER BY HGID
      FOR READ ONLY;
   END
   --Normal Values Entered
   ELSE
   BEGIN
      DECLARE cur CURSOR FOR
         SELECT DISTINCT AgentId, Username COLLATE SQL_Latin1_General_CP1_CI_AS,
                         tblHuntgroup.HGID,
                         tblHuntgroup.[Name] COLLATE SQL_Latin1_General_CP1_CI_AS,
                         FullName COLLATE SQL_Latin1_General_CP1_CI_AS
        FROM tblSupervisorView
        JOIN tblHGViewBridge ON tblSupervisorView.ViewId = tblHGViewBridge.ViewId
        JOIN tblHuntgroup ON tblHGViewBridge.HGID = tblHuntgroup.HGID
        JOIN tblAgentHGBridge ON tblHuntgroup.HGID = tblAgentHGBridge.HGID
        JOIN tblUsers ON tblAgentHGBridge.AgentId = tblUsers.UserId
        WHERE tblSupervisorView.DestroyDate IS NULL
             AND tblHuntgroup.DestroyDate IS NULL
             AND tblUsers.DestroyDate IS NULL
             AND (tblAgentHGBridge.DestroyDate IS NULL OR
                   tblAgentHGBridge.DestroyDate > @FromDate)
             AND tblSupervisorView.[Name] COLLATE SQL_Latin1_General_CP1_CI_AS IN
                  (SELECT * from split(@TargetValue , ','))
             AND tblSupervisorView.SupervisorId = @SupervisorId
          ORDER BY HGID
       FOR READ ONLY;
     END
   END

   --For Huntgroups
   IF (@Target = 'HuntGroup')
   BEGIN
       --Wildcard ALL
       IF (@TargetValue = '*')
       BEGIN
```

```
               DECLARE cur CURSOR FOR
                  SELECT DISTINCT AgentId,
                                 Username COLLATE SQL_Latin1_General_CP1_CI_AS,
                                 tblHuntgroup.HGID,
                                 tblHuntgroup [Name] COLLATE
                                                     SQL_Latin1_General_CP1_CI_AS,
                                 FullName COLLATE SQL_Latin1_General_CP1_CI_AS
                  FROM tblHuntgroup
                  JOIN tblAgentHGBridge ON tblHuntgroup.HGID = tblAgentHGBridge.HGID
                  JOIN tblUsers ON tblAgentHGBridge.AgentId = tblUsers.UserId
                  WHERE tblHuntgroup.DestroyDate IS NULL
                       AND tblUsers.DestroyDate IS NULL
                       AND (tblAgentHGBridge.DestroyDate IS NULL OR
                              tblAgentHGBridge.DestroyDate > @FromDate)
                  ORDER BY HGID
               FOR READ ONLY;
            END
            --Wildcard with a word
            ELSE IF (@StarPos > 0)
            BEGIN
               DECLARE cur CURSOR FOR
                  SELECT DISTINCT AgentId,
                                 Username COLLATE SQL_Latin1_General_CP1_CI_AS,
                                 tblHuntgroup.HGID,
                                 tblHuntgroup.[Name] COLLATE
                                                     SQL_Latin1_General_CP1_CI_AS,
                                 FullName COLLATE SQL_Latin1_General_CP1_CI_AS
                  FROM tblHuntgroup
                  JOIN tblAgentHGBridge ON tblHuntgroup.HGID = tblAgentHGBridge.HGID
                  JOIN tblUsers ON tblAgentHGBridge.AgentId = tblUsers.UserId
                  WHERE tblHuntgroup.DestroyDate IS NULL
                       AND tblUsers.DestroyDate IS NULL
                       AND (tblAgentHGBridge.DestroyDate IS NULL OR
                              tblAgentHGBridge.DestroyDate > @FromDate)
                       AND [Name] COLLATE SQL_Latin1_General_CP1_CI_AS LIKE
                             @TargetValue
                  ORDER BY HGID
               FOR READ ONLY;
            END
            --Normal Values Entered
            ELSE
            BEGIN
               DECLARE cur CURSOR FOR
                  SELECT DISTINCT AgentId,
                                 Username COLLATE SQL_Latin1_General_CP1_CI_AS,
                                 tblHuntgroup.HGID,
                                 tblHuntgroup.[Name] COLLATE
                                                     SQL_Latin1_General_CP1_CI_AS,
                                 FullName COLLATE SQL_Latin1_General_CP1_CI_AS
                  FROM tblHuntgroup
                  JOIN tblAgentHGBridge ON tblHuntgroup.HGID = tblAgentHGBridge.HGID
                  JOIN tblUsers ON tblAgentHGBridge.AgentId = tblUsers.UserId
                  WHERE tblHuntgroup.DestroyDate IS NULL
                       AND tblUsers.DestroyDate IS NULL
                       AND (tblAgentHGBridge.DestroyDate IS NULL OR
                              tblAgentHGBridge.DestroyDate > @FromDate)
                       AND [Name] COLLATE SQL_Latin1_General_CP1_CI_AS IN (SELECT *
                            from split(@TargetValue , ','))
                  ORDER BY HGID
               FOR READ ONLY;
            END
         END

         --For Agents
         IF (@Target = 'CSR')
         BEGIN
            --Wildcard ALL
            IF (@TargetValue = '*')
            BEGIN
               DECLARE cur CURSOR FOR
                  SELECT DISTINCT AgentId, Username, tblHuntgroup.HGID,
                                 tblHuntgroup.[Name],
                                 FullName COLLATE SQL_Latin1_General_CP1_CI_AS
                  FROM tblUsers
                  JOIN tblAgentHGBridge ON tblAgentHGBridge.AgentID = tblUsers.UserId
                  JOIN tblHuntgroup ON tblHuntgroup.HGID = tblAgentHGBridge.HGID
                  WHERE tblUsers.DestroyDate IS NULL
                       AND (tblAgentHGBridge.DestroyDate IS NULL OR
                              tblAgentHGBridge.DestroyDate > @FromDate)
                  ORDER BY HGID
               FOR READ ONLY;
            END
            --Wildcard with a word
            ELSE IF (@StarPos > 0)
            BEGIN
               DECLARE cur CURSOR FOR
                  SELECT DISTINCT AgentId, Username, tblHuntgroup.HGID,
             tblHuntgroup.[Name],
             FullName COLLATE SQL_Latin1_General_CP1_CI_AS
                  FROM tblUsers
                  JOIN tblAgentHGBridge ON tblAgentHGBridge.AgentID = tblUsers.UserId
                  JOIN tblHuntgroup ON tblHuntgroup.HGID = tblAgentHGBridge.HGID
                  WHERE UserName COLLATE SQL_Latin1_General_CP1_CI_AS LIKE
                        @TargetValue
                       AND tblUsers.DestroyDate IS NULL
                       AND (tblAgentHGBridge.DestroyDate IS NULL OR
tblAgentHGBridge.DestroyDate > @FromDate)
                  ORDER BY HGID
               FOR READ ONLY;
            END
            --Normal Values Entered
            ELSE
            BEGIN
               DECLARE cur CURSOR FOR
                  SELECT DISTINCT AgentId, Username, tblHuntgroup.HGID,
             tblHuntgroup.[Name],
             FullName COLLATE SQL_Latin1_General_CP1_CI_AS
                  FROM tblUsers
                  JOIN tblAgentHGBridge ON tblAgentHGBridge.AgentID = tblUsers.UserId
                  JOIN tblHuntgroup ON tblHuntgroup.HGID = tblAgentHGBridge.HGID
                  WHERE UserName COLLATE SQL_Latin1_General_CP1_CI_AS IN (SELECT *
   from split(@TargetValue , ','))
                       AND tblUsers.DestroyDate IS NULL
                       AND (tblAgentHGBridge.DestroyDate IS NULL OR
tblAgentHGBridge.DestroyDate > @FromDate)
                  ORDER BY HGID
               FOR READ ONLY;
            END
         END

         OPEN cur;

         --Loop through all agents
         DECLARE @AgentId bigint;
         DECLARE @HuntgroupId bigint;
         DECLARE @AgentName varchar(50);
         DECLARE @HuntgroupName varchar(50);
         DECLARE @FullAgentName varchar(60);
         DECLARE @StoreAgentId bigint;
         DECLARE @StoreHuntgroupId bigint;
         DECLARE @fetchStatus int;

         FETCH NEXT FROM cur INTO @AgentId, @AgentName, @HuntgroupId, @HuntgroupName,
                                  @FullAgentName;
         SET @fetchStatus = @@FETCH_STATUS;

         --Loop through all agents
         WHILE (0 = 0)
         BEGIN
            SET @StoreAgentId = @AgentId;
            SET @StoreHuntgroupId = @HuntgroupId;
            --Loop per huntgroup
            WHILE (@StoreAgentId = @AgentId AND @fetchStatus = 0)
            BEGIN
               --Initialise the variables
               SET @OtherTime = 0;
               SET @RingTime = 0;
               SET @Outbound  = 0;
               SET @Inbound  = 0;
               SET @Internal  = 0;
               SET @BusyNotAvailableTime  = 0;
               SET @ACWTime  = 0;
               SET @HoldTime  = 0;
               SET @OffHookTime  = 0;
               SET @HGEnabled = 0;

               --Get Huntgroup Enabled
               EXEC dbo.spAgentSummaryReportHGEnabled @AgentId, @HuntgroupId,
                                                      @FromDate, @ToDate, @DaysOfWeek,
                                                      @HGEnabled OUTPUT, 0;

               --Get Huntgroup Related Totals
               EXEC dbo.spAgentSummaryReportHGTotals @AgentId, @HuntgroupId,
                                                     @FromDate, @ToDate, @DaysOfWeek,
                                                     @HoldTime OUTPUT,
                                                     @RingTime OUTPUT,
                                                     @Inbound OUTPUT;

                EXEC dbo.spAgentSummaryReportNonHGTotals @StoreAgentId, @FromDate,
                                                        @ToDate, @DaysOfWeek,
                                                        @Outbound OUTPUT,
                                                        @BusyNotAvailableTime OUTPUT,
                                                        @ACWTime OUTPUT,
                                                        @HoldTime OUTPUT,
                                                        @OffHookTime OUTPUT,
                                                        @Internal OUTPUT,
                                                        @OtherTime OUTPUT

               --Setup Initial Agent In Temporary Table
               INSERT INTO #agentSummary
               VALUES (
                  @AgentId,
                        @FullAgentName,
                   @HuntgroupId,
                   @HuntgroupName,
                   @OtherTime,
                   @RingTime,
                   @Outbound,
                   @Inbound,
                   @Internal,
                   @BusyNotAvailableTime,
                   @ACWTime,
                   @HoldTime,
                   @OffHookTime,
                   @HGEnabled
                   );

                FETCH NEXT FROM cur INTO @AgentId, @AgentName, @HuntgroupId,
     @HuntgroupName, @FullAgentName;
                SET @fetchStatus = @@FETCH_STATUS;
             END

             IF (@fetchStatus <> 0)
                BREAK;
          END

          SELECT * FROM #agentSummary ORDER BY HuntgroupName, AgentName;

          --Clean Up
          CLOSE cur;
          DEALLOCATE cur;
          DROP TABLE #agentSummary;
       END
```

3.3.2 C# Code

The following example C# program shows how to execute the spAgentSummaryReport 36 stored procedure to obtain an Agent Summary Report. The parameters are set using the CSR Target for Agent Extn872, calls between 9:00 and 17:00 including Saturday and Sunday, date range from the first time calls were recorded in the database until now. The SupervisorID value is ignore for CSR targets, it is only used for Supervisor Views target.

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace ConsoleApplication1
{
   class Program
   {
      static void Main()
      {
         try
         {
            SqlConnection connection = new SqlConnection("Data Source=localhost\\SQLEXPRESS;Initial
Catalog=AvayaSBCCRT;uid=username;pwd=password");
            using (connection)
            {
               SqlCommand command =
                  new SqlCommand( "spAgentSummaryReport",
                                  connection);
               using (command)
               {
                  command.CommandType = CommandType.StoredProcedure;

                  SqlParameter param = command.Parameters.Add("Target",
                                       SqlDbType.NVarChar);
                  param.Direction = ParameterDirection.Input;
                  param.Value = "CSR";

                  param = command.Parameters.Add("TargetValue",
                                                 SqlDbType.NVarChar);
                  param.Direction = ParameterDirection.Input;
                  param.Value = "Extn872";

                  param = command.Parameters.Add("IncludeSaturday",
                                                 SqlDbType.Bit);
                  param.Direction = ParameterDirection.Input;
                  param.Value = true;

                  param = command.Parameters.Add("IncludeSunday",
                          SqlDbType.NVarChar);
                  param.Direction = ParameterDirection.Input;
                  param.Value = true;

                  param = command.Parameters.Add("FromDate",
                                                 SqlDbType.DateTime);
                  param.Direction = ParameterDirection.Input;
                  param.Value = DateTime.Now.AddDays(-1);

                  param = command.Parameters.Add("ToDate",
                                                 SqlDbType.DateTime);
                  param.Direction = ParameterDirection.Input;
                  param.Value = DateTime.Now;

                  param = command.Parameters.Add("StartTime",
                                                 SqlDbType.NVarChar);
                  param.Direction = ParameterDirection.Input;
                  param.Value = "09:00";

                  param = command.Parameters.Add("EndTime",
                                                 SqlDbType.NVarChar);
                  param.Direction = ParameterDirection.Input;
                  param.Value = "17:00";

                  param = command.Parameters.Add("SupervisorId",
                                                 SqlDbType.Int);
                  param.Direction = ParameterDirection.Input;
                  param.Value = 2;

                  connection.Open();

                  SqlDataReader reader = command.ExecuteReader();
                  if (null != reader)
                  {
                     using (reader)
                     {
                        while (reader.Read())
                        {
                           for (int field = 0;
                                field < reader.FieldCount;
                                field++)
                           {
      Console.WriteLine(reader.GetName(field)
                                              + ": "
                                              + reader[field]);
                           }
                        }
                        while (reader.NextResult())
                        {
                           Console.WriteLine(string.Empty);
                           while (reader.Read())
                           {
                              for (int field = 0;
                                   field < reader.FieldCount;
                                   field++)
                              {
                                Console.WriteLine(reader.GetName(field)
                                               + ": "
                                               + reader[field]);
                              }
                           }
                        }

                        reader.Close();
                     }
                  }

                  connection.Close();
               }
            }
         }
         catch (Exception ex)
         {
            Console.WriteLine(String.Format("Exception: {0}",
                                            ex.Message));
         }
      }
   }
}
```

The returned data set can be used as input to a function that will generate a report (using the Crystal Report toolkit for example) or a function that will format the data and store it to a file (Excel, XML, plain text, etc…).

3.4 Scheduling

The Windows Task Scheduler can be used to schedule a Custom Report Application that does the data mining to create reports. It is used by IP Office Customer Call Reporter for the built-in reports.

For more information on the Task Scheduler, refer http://msdn.microsoft.com/en-us/library/aa383614(v=VS.85).aspx.