IP Office 8.0

IP Office Customer Call Reporter Custom Reporting

- Issue 01a - (Thursday, December 08, 2011)

Notices
While reasonable efforts have been made to ensure that the information in this document is complete and accurate at the time of printing, Avaya assumes no liability for any errors. Avaya reserves the right to make changes and corrections to the information in this document without the obligation to notify any person or organization of such changes.

Documentation disclaimer
Avaya shall not be responsible for any modifications, additions, or deletions to the original published version of this documentation unless such modifications, additions, or deletions were performed by Avaya.

End User agree to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation, to the extent made by End User.

Link disclaimer
Avaya is not responsible for the contents or reliability of any linked Web sites referenced within this site or documentation(s) provided by Avaya. Avaya is not responsible for the accuracy of any information, statement or content provided on these sites and does not necessarily endorse the products, services, or information described or offered within them. Avaya does not guarantee that these links will work all the time and has no control over the availability of the linked pages.

Warranty
Avaya provides a limited warranty on this product. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya's standard warranty language, as well as information regarding support for this product, while under warranty, is available to Avaya customers and other parties through the Avaya Support Web site: http://www.avaya.com/support. Please note that if you acquired the product from an authorized Avaya reseller outside of the United States and Canada, the warranty is provided to you by said Avaya reseller and not by Avaya.

Licenses
THE SOFTWARE LICENSE TERMS AVAILABLE ON THE AVAYA WEBSITE, HTTP://SUPPORT.AVAYA.COM/LICENSEINFO/ ARE APPLICABLE TO ANYONE WHO DOWNLOADS, USES AND/OR INSTALLS AVAYA SOFTWARE, PURCHASED FROM AVAYA INC., ANY AVAYA AFFILIATE, OR AN AUTHORIZED AVAYA RESELLER (AS APPLICABLE) UNDER A COMMERCIAL AGREEMENT WITH AVAYA OR AN AUTHORIZED AVAYA RESELLER. UNLESS OTHERWISE AGREED TO BY AVAYA IN WRITING, AVAYA DOES NOT EXTEND THIS LICENSE IF THE SOFTWARE WAS OBTAINED FROM ANYONE OTHER THAN AVAYA, AN AVAYA AFFILIATE OR AN AVAYA AUTHORIZED RESELLER, AND AVAYA RESERVES THE RIGHT TO TAKE LEGAL ACTION AGAINST YOU AND ANYONE ELSE USING OR SELLING THE SOFTWARE WITHOUT A LICENSE. BY INSTALLING, DOWNLOADING OR USING THE SOFTWARE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE INSTALLING, DOWNLOADING OR USING THE SOFTWARE (HEREINAFTER REFERRED TO INTERCHANGEABLY AS "YOU" AND "END USER"), AGREE TO THESE TERMS AND CONDITIONS AND CREATE A BINDING CONTRACT BETWEEN YOU AND AVAYA INC. OR THE APPLICABLE AVAYA AFFILIATE ("AVAYA").

Avaya grants End User a license within the scope of the license types described below. The applicable number of licenses and units of capacity for which the license is granted will be one (1), unless a different number of licenses or units of capacity is specified in the Documentation or other materials available to End User. "Designated Processor" means a single stand-alone computing device. "Server" means a Designated Processor that hosts a software application to be accessed by multiple users. "Software" means the computer programs in object code, originally licensed by Avaya and ultimately utilized by End User, whether as stand-alone products or pre-installed on Hardware. "Hardware" means the standard hardware originally sold by Avaya and ultimately utilized by End User.

License types
Designated System(s) License (DS). End User may install and use each copy of the Software on only one Designated Processor, unless a different number of Designated Processors is indicated in the Documentation or other materials available to End User. Avaya may require the Designated Processor(s) to be identified by type, serial number, feature key, location or other specific designation, or to be provided by End User to Avaya through electronic means established by Avaya specifically for this purpose.

Copyright
Except where expressly stated otherwise, no use should be made of materials on this site, the Documentation(s) and Product(s) provided by Avaya. All content on this site, the documentation(s) and the product(s) provided by Avaya including the selection, arrangement and design of the content is owned either by Avaya or its licensors and is protected by copyright and other intellectual property laws including the sui generis rights relating to the protection of databases. You may not modify, copy, reproduce, republish, upload, post, transmit or distribute in any way any content, in whole or in part, including any code and software. Unauthorized reproduction, transmission, dissemination, storage, and or use without the express written consent of Avaya can be a criminal, as well as a civil, offense under the applicable law.

Third Party Components
Certain software programs or portions thereof included in the Product may contain software distributed under third party agreements ("Third Party Components"), which may contain terms that expand or limit rights to use certain portions of the Product ("Third Party Terms"). Information regarding distributed Linux OS source code (for those Products that have distributed the Linux OS source code), and identifying the copyright holders of the Third Party Components and the Third Party Terms that apply to them is available on the Avaya Support Web site: http://support.avaya.com/Copyright.

Preventing toll fraud
"Toll fraud" is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or is not working on your company's behalf). Be aware that there can be a risk of toll fraud associated with your system and that, if toll fraud occurs, it can result in substantial additional charges for your telecommunications services.

Avaya fraud intervention
If you suspect that you are being victimized by toll fraud and you need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1-800-643-2353 for the United States and Canada. For additional support telephone numbers, see the Avaya Support Web site: http://support.avaya.com
Suspected security vulnerabilities with Avaya products should be reported to Avaya by sending mail to: securityalerts@avaya.com.

Trademarks
Avaya and Aura are trademarks of Avaya, Inc.
The trademarks, logos and service marks ("Marks") displayed in this site, the documentation(s) and product(s) provided by Avaya are the registered or unregistered Marks of Avaya, its affiliates, or other third parties. Users are not permitted to use such Marks without prior written consent from Avaya or such third party which may own the Mark. Nothing contained in this site, the documentation(s) and product(s) should be construed as granting, by implication, estoppel, or otherwise, any license or right in and to the Marks without the express written permission of Avaya or the applicable third party. Avaya is a registered trademark of Avaya Inc. All non-Avaya trademarks are the property of their respective owners.

Downloading documents
For the most current versions of documentation, see the Avaya Support Web site: http://www.avaya.com/support

Contact Avaya Support
Avaya provides a telephone number for you to use to report problems or to ask questions about your product. The support telephone number is 1-800-242-2121 in the United States. For additional support telephone numbers, see the Avaya Web site: http://www.avaya.com/support

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 7.0.

- **! 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

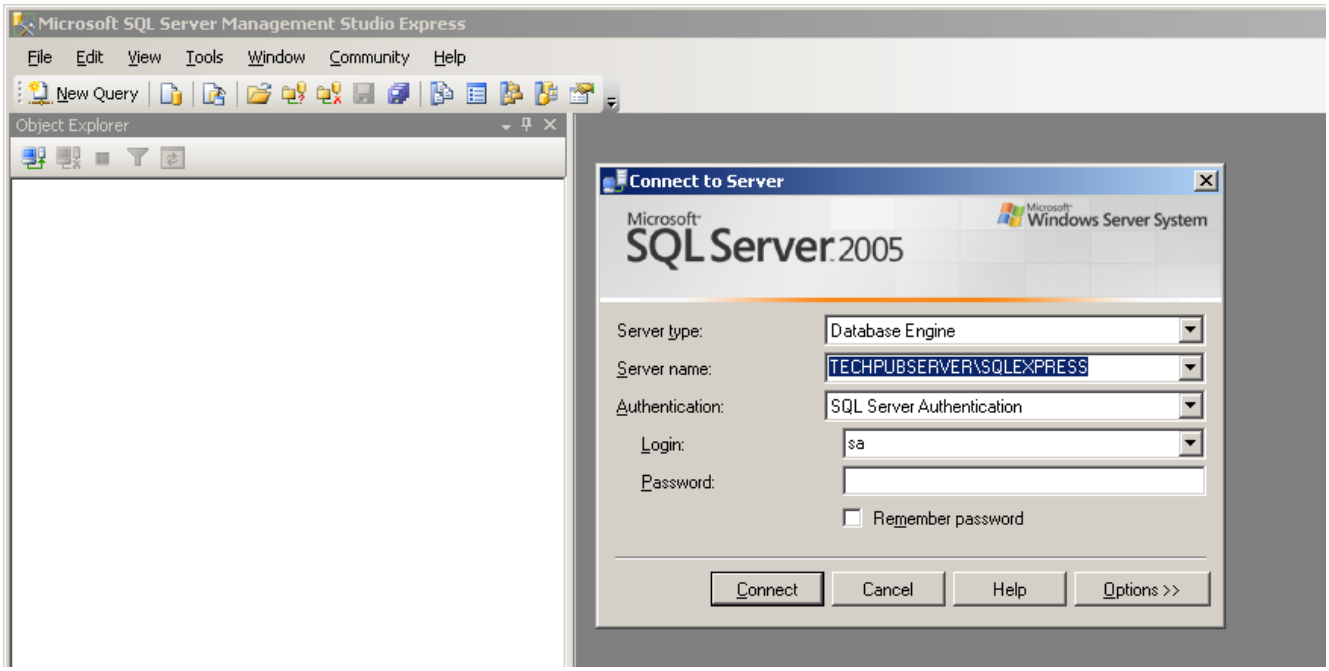
If remote access to the database is needed, certain TCP/IP protocols and the SQL browser service need to be enabled on the SQL Server PC. In addition, firewall rules may need to be modified. This is described in the document, http://support.microsoft.com/kb/914277 and is not needed for local access which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

1.3 Management Studio Express

The IP Office Customer Call Reporter database can be "viewed" using the Management Studio Express. 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.

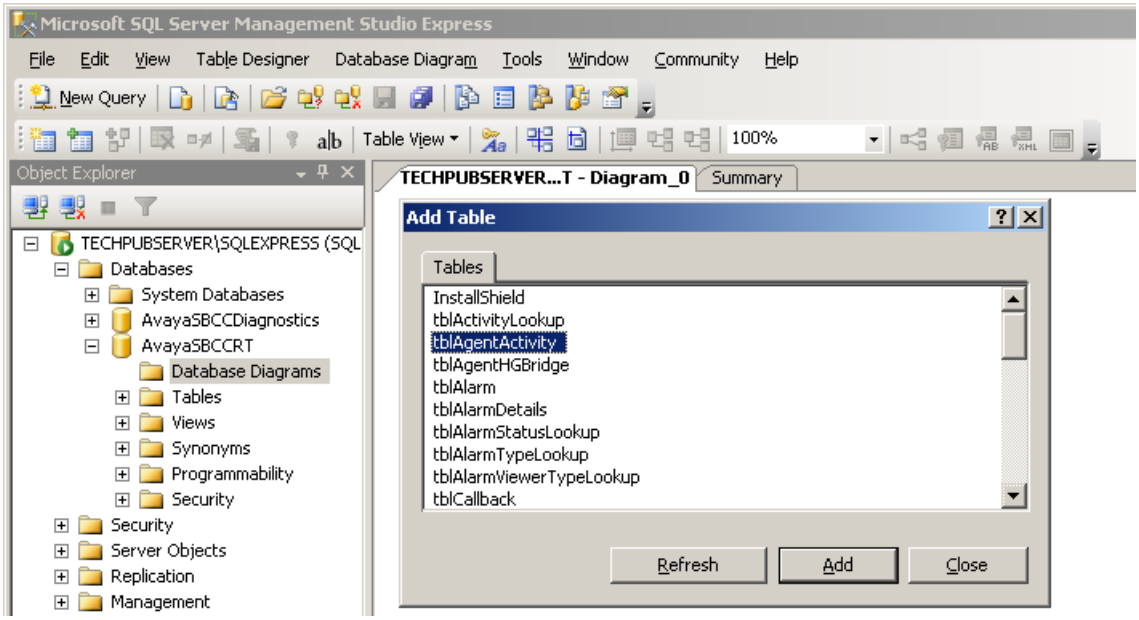
- **! WARNING**
  The design of custom reports using the data provided in the IP Office Customer Call Reporter database is the property of the designer and all support associated with it is to be provided by that designer. Removing or modifying any of the Tables, Relations, Stored Procedures, Functions 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.



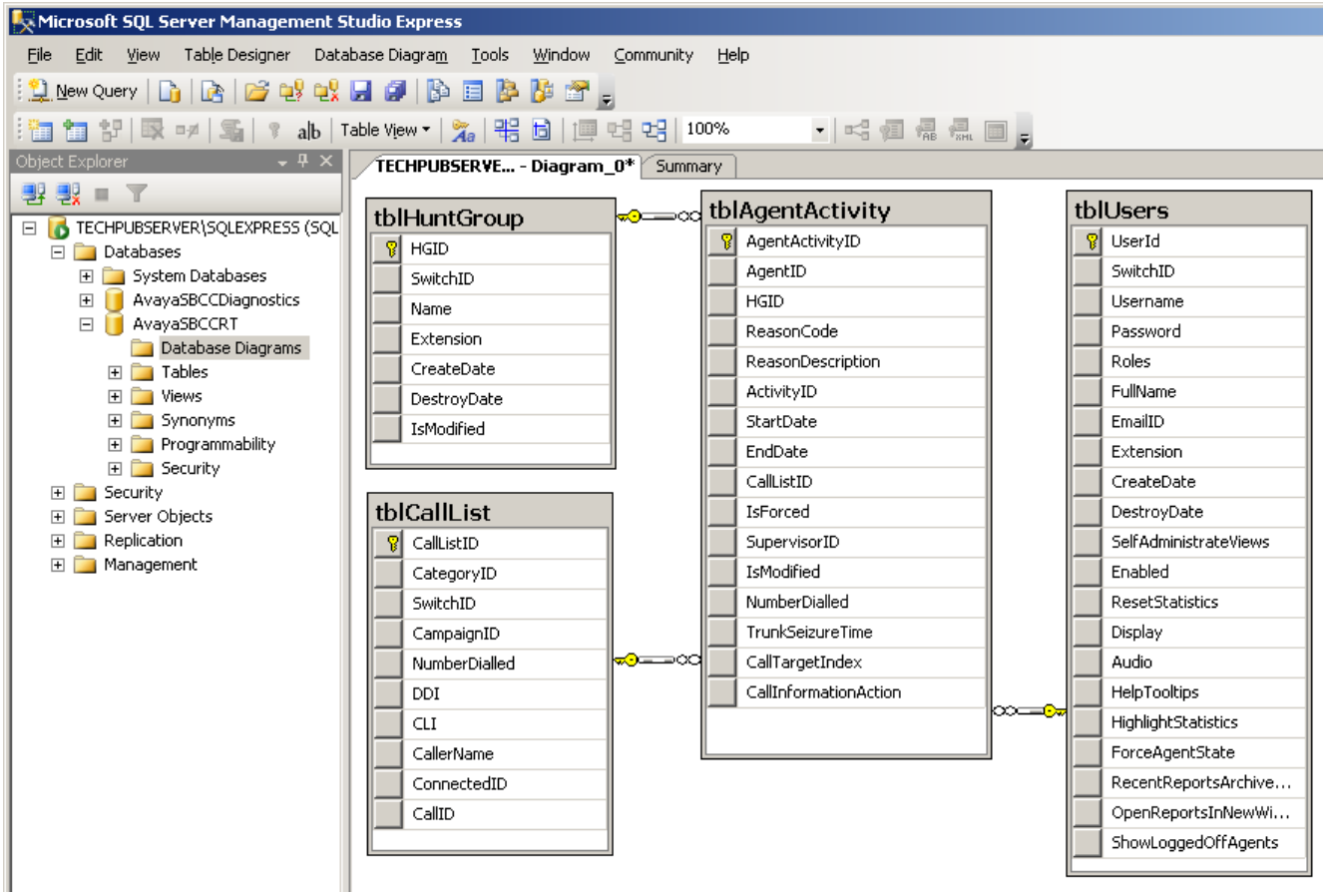
Database Diagrams

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.

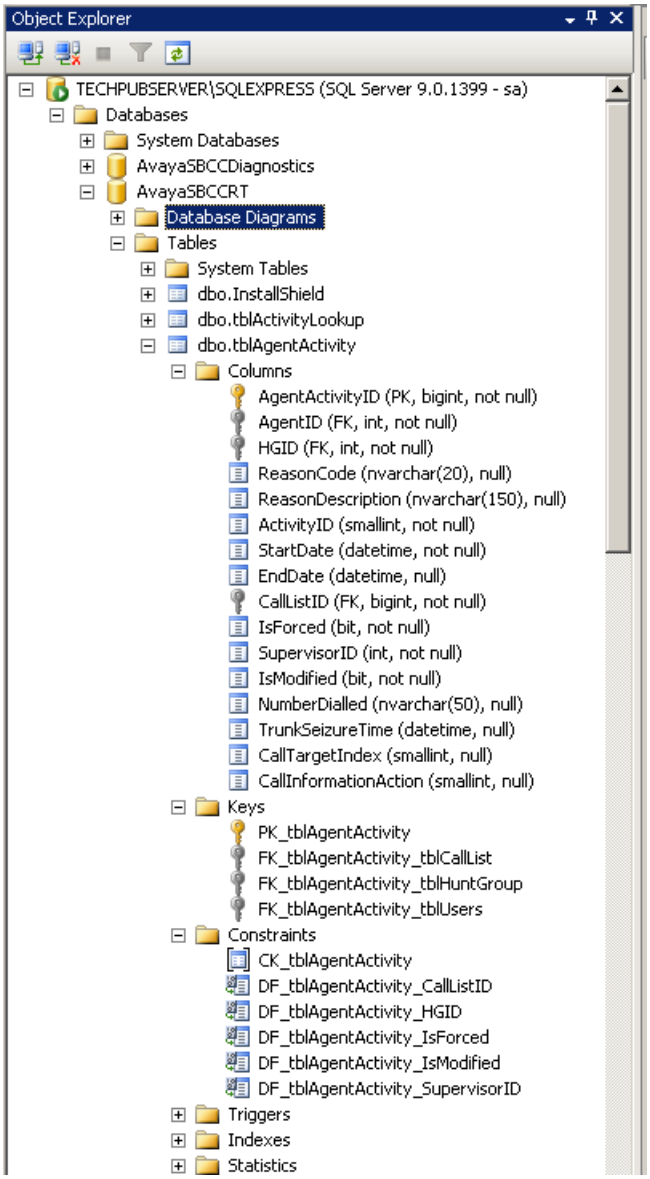


Then, a right click on the added table and a request to add related tables can be made. That will show the relationship between tables. Different views can be selected (table names only, with keys only, with column definitions, etc).



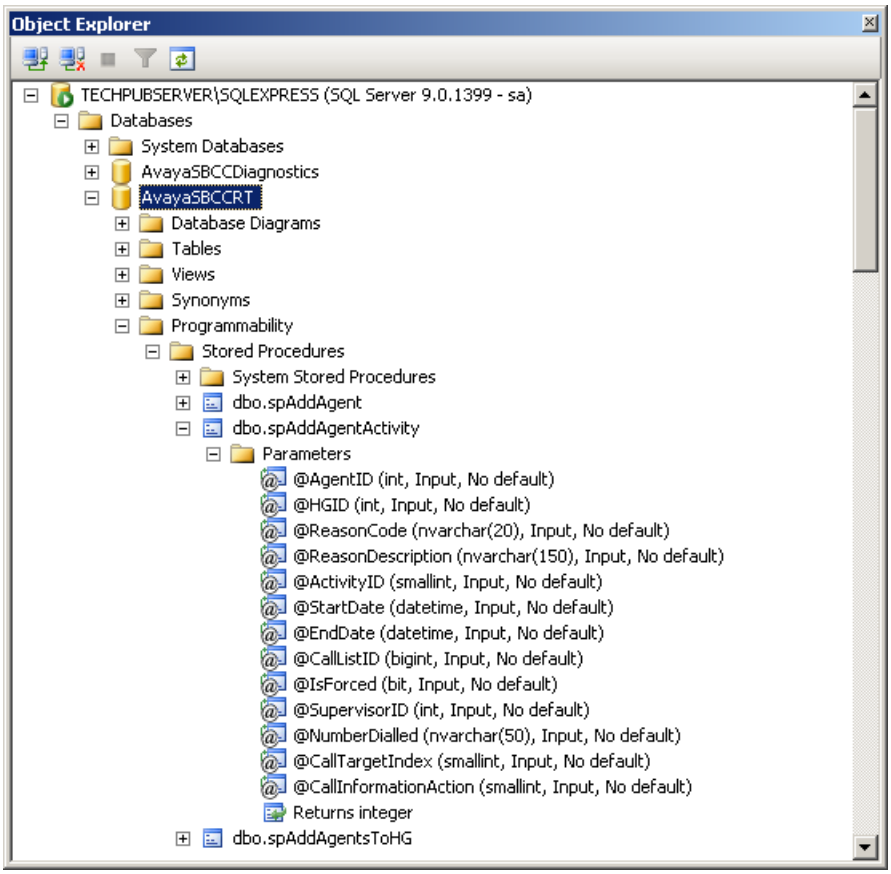
Tables

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



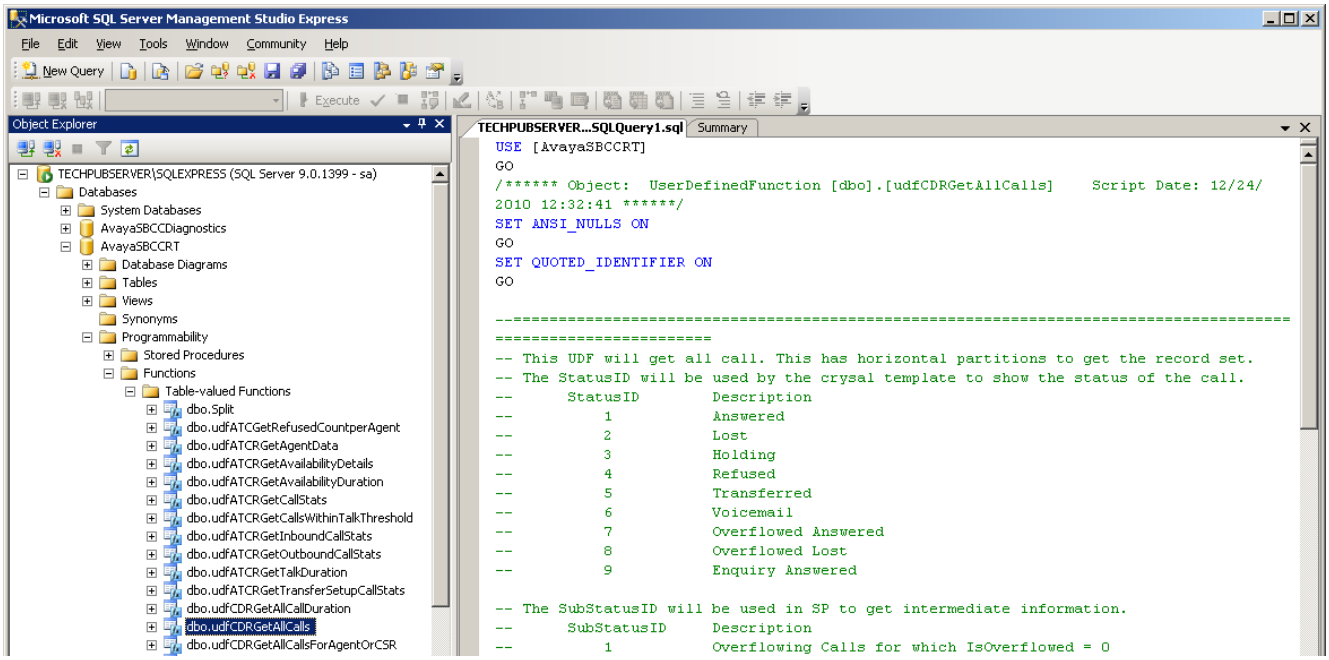
Stored Procedures

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



Functions

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



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.

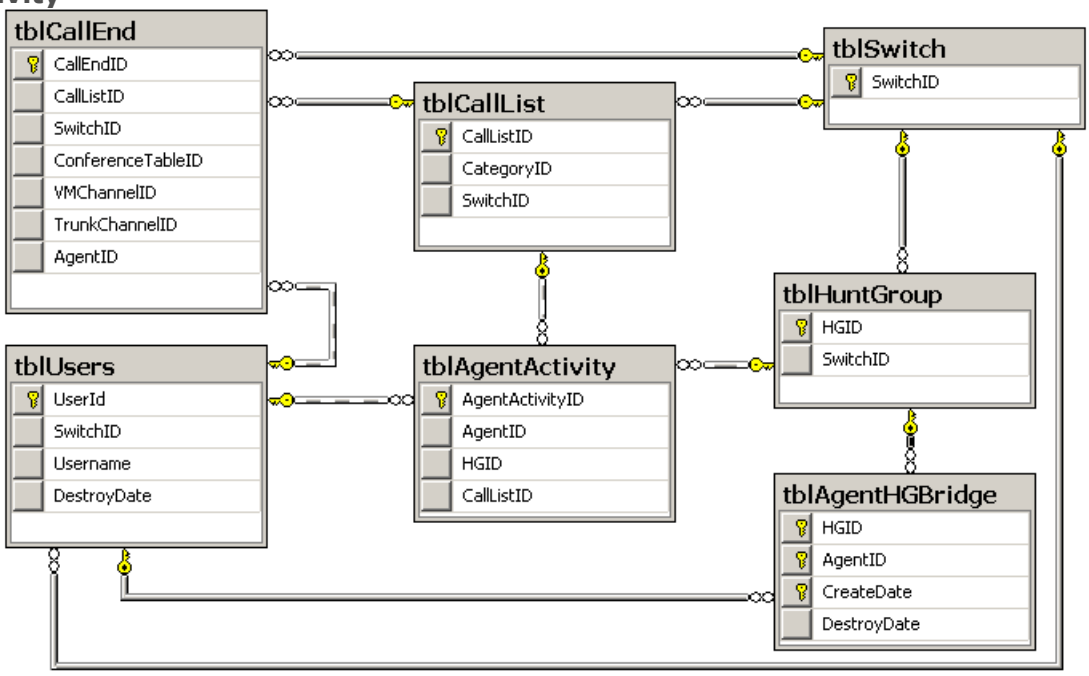
2.1 Database Tables

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 20	**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 20	**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 18	**Master**	Stores list of the built in basic reports.
tblReportParametersScheduleLookup 18	**Lookup**	Stores the list of Schedule options.

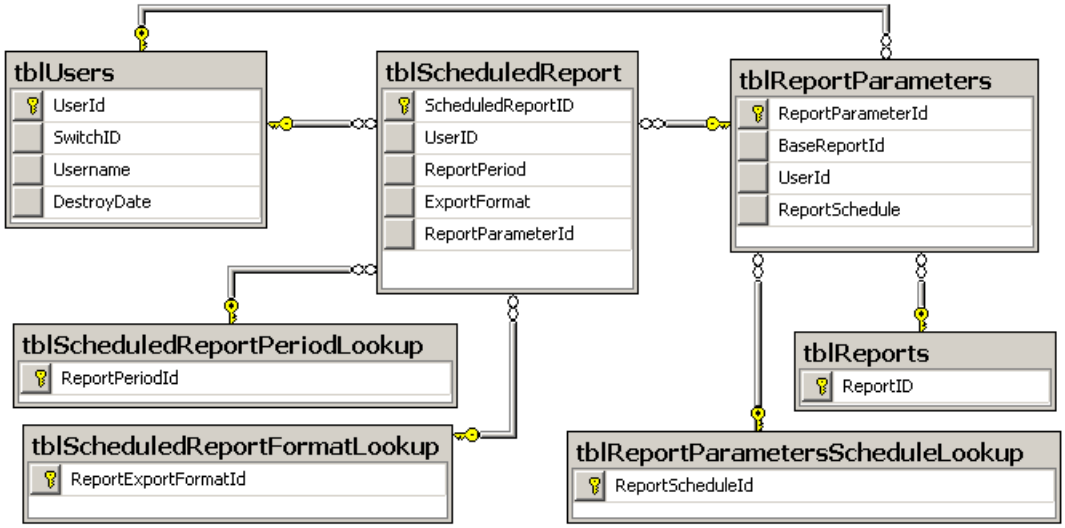
Not all tables are shown in the following diagrams, only those that are used for reports. For a full database diagram, use the Management Studio Express tool. For the purpose of clarity, the diagram is split in two parts. The first part is about call activity and the second is about the reporting. The 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 20 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 21 (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 21 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 20
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 21. Foreign key to tblCategoryLookup
SwitchID	int	4	No	No	Stores the Switch / IPOffice ID. Foreign key to tblSwitch 20.
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.
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.

Column	Data Type	Len	Null able	Identity	Remarks
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 20.
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 18.
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 20.
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.

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.

Column	Data Type	Len	Nullable	Identity	Remarks
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 20
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.

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.
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.

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 20.
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.
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	0	All
	9	Shifts
	10	Lunch
	11	Breaks
	12	Talk Time
	13	Performance
	14	Calls
Call Detail Report	0	All
	1	Answered
	2	No answer
	3	Overflowed Lost
	4	Overflowed Answered
	5	Transferred
	6	Lost Calls
	7	Routed to voicemail
Call Summary Report	0	All

2.1.14.4 tblReportGroups

Report	GroupId	Description
Agent Summary Report	5	QUEUE
Agent Time Card	3	DAY
	4	WEEK
	5	AGENT
Call Detail Report	1	UNGROUPED
	2	HOUR
	3	DAY
	4	WEEK
	5	QUEUE
	6	AGENT
	7	CLI
	8	DDI
	9	ACCOUNT CODE
Call Summary Report	1	UNGROUPED
	2	HOUR
	3	DAY
	4	WEEK
	5	QUEUE
	6	AGENT
	7	CLI
	8	DDI
	9	ACCOUNT CODE
Voicemail Report	1	UNGROUPED
	2	HOUR
	3	DAY
	4	WEEK
	7	CLI
	8	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	0	Queue
	1	View
	2	Agent
Agent Time Card	2	Agent
Call Detail Report	0	Queue
	1	View
	2	Agent
	3	DDI
	4	CLI
	5	Account code
Call Summary Report	0	Queue
	1	View
	2	Agent
	3	DDI
	4	CLI
	5	Account code
Trace Report	2	Agent
	4	CLI
	8	Call reference
Voicemail Report	7	Voicemail

2.1.14.8 tblStateGroupLookup

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 Express 9.

S. No.	Procedure
1	spAddAgent
2	spAddAgentActivity
3	spAddAgentsToHG
4	spAddAlarm
5	spAddAlarmDetails
6	spAddCallEnd
7	spAddCallList
8	spAddConference
9	spAddExtension
10	spAddExtensionsToHG
11	spAddHG
12	spAddRTRequest
13	spAddRTStat
14	spAddSwitch
15	spAddSwitchActivity
16	spAddSwitchWithPendingStatus
17	spAddTrunkChannel
18	spAddTrunkGroup
19	spAddTrunkGroupBusyTime
20	spAddVMChannel
21	spAddVMGroup
22	spAddVMSelection
23	spAgentSummaryReport
24	spAgentSummaryReportHGEnabled
25	spAgentSummaryReportHGTotals
26	spAgentSummaryReportNonHGTotals
27	spAgentTimeCardReport
28	spAlarmReport
29	spCallDetailReport
30	spCallDetailReportForAccountCode
31	spCallDetailReportForAgentORCSR
32	spCallDetailReportForCLIDDI
33	spCallDetailReportForHuntGroup
34	spCallDetailReportForView
35	spCallSummaryReport
36	spCallSummaryReportForAccountCode
37	spCallSummaryReportForCLI
38	spCallSummaryReportForCSR
39	spCallSummaryReportForDDI
40	spCallSummaryReportForHuntGroup
41	spCallSummaryReportForView
42	spClearCache
43	spClearConference
44	spDatabaseMonitorDeleteAll
45	spDatabaseMonitorDeleteCallData
46	spDatabaseMonitorDeleteHuntTrunkGroupsSwitches
47	spDatabaseMonitorDeleteOldestPercentageCallData
48	spDatabaseMonitorDeleteSafe
49	spDatabaseMonitorDeleteUsersViewsReports
50	spDatabaseMonitorGetSize
51	spDatabaseMonitorMain
52	spDatabaseMonitorRebuildIndexes

S. No.	Procedure
107	spManagementServiceDestroyUser
108	spManagementServiceDestroyWallboardUser
109	spManagementServiceGetActivities
110	spManagementServiceGetAllCSRs
111	spManagementServiceGetAllSignOnTimeOut
112	spManagementServiceGetAllSignOnTimeOutRole
113	spManagementServiceGetAllSupervisors
114	spManagementServiceGetAllSystemSettings
115	spManagementServiceGetAllWallboards
116	spManagementServiceGetAllWallboardsUnlocked
117	spManagementServiceGetAllWallboardUsers
118	spManagementServiceGetCSR
119	spManagementServiceGetCSRsInHuntGroup
120	spManagementServiceGetDashboardGoal
121	spManagementServiceGetDashboardPanes
122	spManagementServiceGetHuntGroups
123	spManagementServiceGetHuntGroupsForCSR
124	spManagementServiceGetHuntGroupsSupervisor
125	spManagementServiceGetHuntGroupStates
126	spManagementServiceGetHuntGroupsView
127	spManagementServiceGetMonitoringAlarms
128	spManagementServiceGetPassword
129	spManagementServiceGetSchemaVersion
130	spManagementServiceGetSignOnTimeOut
131	spManagementServiceGetSignOnTimeOutPerSession
132	spManagementServiceGetSignOnTimeOutPerUserSession
133	spManagementServiceGetStatLookup
134	spManagementServiceGetStatLookupView
135	spManagementServiceGetStatParameters
136	spManagementServiceGetSuperAdmin
137	spManagementServiceGetSupervisor
138	spManagementServiceGetSupervisorViews
139	spManagementServiceGetSupervisorViewSettings
140	spManagementServiceGetSystemSetting
141	spManagementServiceGetTimeOut
142	spManagementServiceGetTrunkGroups
143	spManagementServiceGetTrunkGroupsSupervisor
144	spManagementServiceGetUserBase
145	spManagementServiceGetUserId
146	spManagementServiceGetUserRoles
147	spManagementServiceGetViewSettingsHGStateThreshold
148	spManagementServiceGetViewSettingsStateThreshold
149	spManagementServiceGetWallboard
150	spManagementServiceGetWallboardLock
151	spManagementServiceGetWallboardsSupervisor
152	spManagementServiceListAllSwitches
153	spManagementServiceResetPassword
154	spManagementServiceResetPasswordByUsername
155	spManagementServiceRollSignOn
156	spManagementServiceSetDashboardGoal
157	spManagementServiceSetDashboardPane

S. No.	Procedure
53	spDeleteAlarm
54	spDeleteAlarmById
55	spDeleteMaintenance
56	spDeleteSavedReport
57	spEndAgentActivity
58	spGetActiveAgentHG
59	spGetActiveAgents
60	spGetActiveExtensionHG
61	spGetActiveExtensions
62	spGetActiveHGs
63	spGetActiveSwitches
64	spGetActiveTrunks
65	spGetActiveVMs
66	spGetAlarms
67	spGetAllLastRunTasks
68	spGetAllSupervisorViews
69	spGetAllSwitches
70	spGetAllTargets
71	spGetCategoryLookup
72	spGetCLIs
73	spGetLastRunReports
74	spGetMaintenanceProperties
75	spGetMaintenanceTasks
76	spGetProcessId
77	spGetReportInfo
78	spGetReportParameters
79	spGetSavedReportParameters
80	spGetScheduledReports
81	spGetScheduleProperties
82	spGetStatGroupLookup
83	spGetStatLookup
84	spGetSubjects
85	spGetTargetFilterInAndLikeLiterals
86	spGetTargetFilters
87	spGetTargetList
88	spGetViewThresholdsForHGCollection
89	spGraphReport
90	spInitializeSPInput
91	spKillProcessId
92	spLastStatsReset
93	spListReports
94	spManagementServiceChangePassword
95	spManagementServiceCreateSignOn
96	spManagementServiceCreateSuperAdmin
97	spManagementServiceCreateSupervisor
98	spManagementServiceCreateWallboard
99	spManagementServiceCreateWallboardUser
100	spManagementServiceDeleteHuntGroupSupervisorBridge
101	spManagementServiceDeleteHuntGroupView
102	spManagementServiceDeleteSignOn
103	spManagementServiceDeleteSwitch
104	spManagementServiceDeleteTrunkGroupSupervisorBridge
105	spManagementServiceDestroyStatLookupViewBridge
106	spManagementServiceDestroySupervisor

S. No.	Procedure
158	spManagementServiceSetHuntGroupSupervisorBridge
159	spManagementServiceSetHuntgroupViewBridge
160	spManagementServiceSetStatLookupViewBridge
161	spManagementServiceSetStatParameters
162	spManagementServiceSetSupervisorView
163	spManagementServiceSetSupervisorViewSettings
164	spManagementServiceSetSystemSetting
165	spManagementServiceSetTrunkGroupSupervisorBridge
166	spManagementServiceSetViewSettingsHGStateThreshold
167	spManagementServiceSetViewSettingsStateThreshold
168	spManagementServiceSetWallboardLock
169	spManagementServiceUpdateCSR
170	spManagementServiceUpdateSuperAdmin
171	spManagementServiceUpdateSupervisor
172	spManagementServiceUpdateWallboard
173	spManagementServiceVerifySuperAdmin
174	spManagementServiceVerifyWallboardUser
175	spRemoveAgent
176	spRemoveAgentsFromHG
177	spRemoveExtension
178	spRemoveExtensionsFromHG
179	spRemoveHG
180	spRemovePendingSwitch
181	spRemoveRTRequest
182	spRemoveSwitch
183	spRemoveTrunkGroup
184	spRemoveVMGroup
185	spSaveLastRunReport
186	spSaveReportParameters
187	spSaveScheduleProperties
188	spSwitchDisconnectivityDetail
189	spSystemGetAnsweredCall
190	spSystemGetLostCall
191	spSystemUpdateAnsweredCall
192	spSystemUpdateLostCall
193	spTraceReport
194	spUpdateAgentActivity
195	spUpdateAlarmDetails
196	spUpdateAlarmStatus
197	spUpdateAlarmThresholds
198	spUpdateCallList
199	spUpdateMaintenanceProperties
200	spUpdateReportParameters
201	spUpdateScheduleProperties
202	spUpdateStatValue
203	spUpdateSwitchConnection
204	spUpdateSwitchConnectionStatus
205	spUpdateSwitchDetails
206	spUpdateSwitchParameters
207	spUpdateSwitchWithPendingStatus
208	spVoiceMailReport
209	spWallBoardMessageAddMessage
210	spWallBoardMessageDeleteMessage
211	spWallBoardMessageGetCurrentMessages
212	spWallBoardMessageGetMessage
213	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 Express[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	udfCDRGetAnsweredCalls
17	udfCDRGetHeldCalls
18	udfCDRGetHeldDuration
19	udfCDRGetLostCalls
20	udfCDRGetMainCallView
21	udfCDRGetMainCallViewForAccountCode
22	udfCDRGetMainCallViewForAgentORCSR
23	udfCDRGetMainCallViewForCLIDDI
24	udfCDRGetMainCallViewForHuntGroups
25	udfCDRGetMainCallViewForOverFlowedHuntGroups
26	udfCDRGetMainCallViewForOverFlowedHuntGroups1
27	udfCDRGetMainCallViewForOverFlowedViews
28	udfCDRGetMainCallViewForViews
29	udfCDRGetOverflowedAnsweredCalls
30	udfCDRGetOverflowedLostCalls
31	udfCDRGetOverflowingCalls
32	udfCDRGetOverflowingDetails
33	udfCDRGetQueueTime
34	udfCDRGetRefusedCalls
35	udfCDRGetRoutedToVMCSR

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

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 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.