

Contact Q System Specification

System release v1 (pre-release draft specification)

Braxtel EMEA Office

**Foxhall Business Centre
Foxhall Road
Nottingham NG7 6LH**

Corporate Head Office

**99 Washington St
Melrose
MA 02176
USA**

Notice

The specifications and information regarding the products in this system specification are subject to change without notice. All statements, information and recommendations are believed to be correct at the time of compilation but are presented without warranty of any kind, express or implied. Users of this information must take full responsibility for their application of any products

The software license and limited warranty for the accompanying product are set forward in the information packet that shipped with the product and incorporated herein by this reference. If you are unable to locate the software license or limited warranty, contact your Braxtel representative for a copy.

Notwithstanding any other warranty herein, all document files and software of these suppliers are provided "As Is" with all faults. Braxtel and the suppliers of its products disclaim all warranties, expressed or implied, including without limitation, those of merchantability, fitness for a particular purpose and non-infringement or arising from a course of dealing, usage or trade practice.

In no event shall Braxtel or its supplier be liable for any indirect, special, consequential or incidental damages including without limitation lost profits or loss or damages to data arising out of the use of inability to user this document, even if Braxtel or its suppliers have been advised of the possibility of such damages.

Interfaces Supported in this Version

Inter-Connectivity with ContactQ (Using SIP Trunking)

The ContactQ platform appliance in release v1 is designed to be a SIP media contact centre application server. In order to enable call signalling and voice call exchange between the ContactQ media processing gateway and a PBX, SIP signalling trunks endpoints are required to be configured for inbound and outbound calls.

Adjunct IP Gateways can be utilised to enable TDM integration where required and allow advantage to be taken from IP-based services and end points. Equally some third party gateways can support supplementary services over SIP-to-Q.SIG and SIP-to-DPNSS.

ContactQ uses a B2BUA transaction agent model facilitating media stream dependent call centre functionality in one deployment including supervisor call intercepting (passive, active, whisper, or steal/control), playing DTMF tones, conferencing, call recording (logging), and comprehensive call control features. The ContactQ media gateway acts as a user agent for incoming calls and sets up new call sessions to agents/users, actively remaining within the media (RTP) stream.

SIP trunking between a PBX and the ContactQ media gateway is established either through a registration free call level authentication method or through a SIP client connection. The registration free method is where the incoming trunk simply sends the digits dialled in an INVITE message, (the call setup message).

SIP Proxy parameters

Host – Proxy host address

Port – Proxy host port

Username - <username[@realm]> : If ContactQ is accepting SIP INVITE requests from a remote SIP client, this field specifies the user name for authentication.

Password - If ContactQ is accepting SIP INVITE requests from a remote SIP client, this field specifies the user associated password for authentication.

From User - <from_ID> : Specify user to put in "from" instead of caller id (overrides the callerid) when placing calls _to_ peer (another SIP proxy).

From Domain – sets default From : Domain in SIP messages

DTMF Mode – rfc2833, info, band, specifies how DTMF signalling is handled

Insecure – very / no : Specifies how to handle connections with SIP peers. Default very (does not authenticate connections).

NAT – yes / no : This parameter changes the behaviour when behind a firewall
SIP Reinvites - specifies if the client supports SIP reinvites

The second method for connecting the platform on a domain level to another SIP trunking endpoint is through a client registration setting (register with a SIP provider) using an appropriately formatted connection string:

user[:secret[:authuser]]@host[:port][/extension]

Registrations to SIP servers use the REGISTER method.

Registration parameters for SIP Clients

user is the user id for this SIP server

authuser is the optional authorization user for the SIP server

secret is the user's password

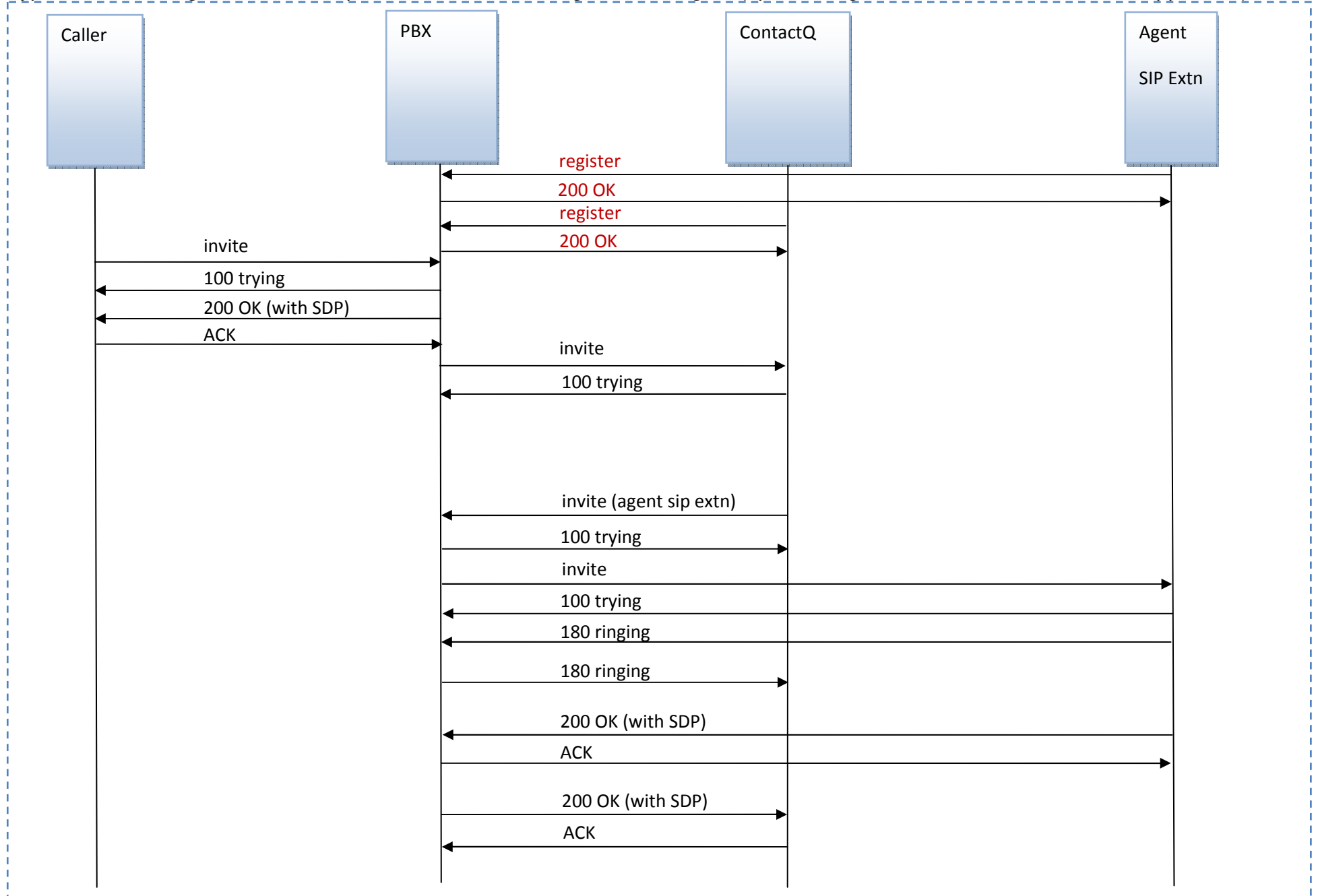
host is the domain or host name for the SIP server.

port send the register request to this port at host. Defaults to 5060

/extension is the contact extension. This extension as specified is included in the contact header in the SIP REGISTER message. The contact extension is used by remote SIP server when it needs to route a call to ContactQ.

Where connections are initiated outwards to a PBX from the ContactQ media gateway on a ContactQ outbound SIP trunk using proxy settings, the signalling uses an INVITE message. For connected ContactQ call centre calls, media control remains within the application server.

Typical Call Routing User Case Sequence from caller through ACD to Agent (optional registrations shown in red where applicable)



CDR

CDR collection is an application server process in ContactQ which logs call and user activity through an event publishing service which is a transaction caching agent for system events. ContactQ can be configured to log call detail records (CDRs) and agent detail records (ADRs) to a SQLite domain database, equally. The CDR cache can be set to be periodically transformed into a higher value database reporting data set using a domain level data transformation service. The same process can be configured to purge the transaction cache. Application scripts driving the system state when modified and customised

CDR Table Schema definition

The Publisher.DB - CDR table represents the primary persistence layer for all call CDR records / events in the system. Three types of record are currently written depending on the Contact map where contacts are registered entities including the IVR, ACD and Agent Manager Application server processes.

Field Name	Data Type	Description	Default
urn	INT(8)	Unique reference number – Auto increment	Next URN
payloadurn	INT(8)	Reference to call payload table object URN	N/A
callid	VARCHAR(40)	Unique call ID	
contact	VARCHAR(40)	Contact ID	
from	VARCHAR(40)	Call originator	
to	VARCHAR(40)	Call destination, ACD Queue, IVR application	
originalto	VARCHAR(40)	Original Call Destination, DNIS for ACD	
routename	VARCHAR(40)	Contact Route Name	
timeplan	VARCHAR(40)	Contact Time Plan	
mediatype	VARCHAR(40)	Call Media Type	
mediaid	VARCHAR(40)	Call Media ID	
locale	VARCHAR(10)	Locale	en_US
recurl	VARCHAR(80)	Record URL	
agent	VARCHAR(82)	Agent handling this leg of the call, NULL if not	NULL

		applicable	
disposition	VARCHAR(40)	Call Disposition	
reason	VARCHAR(40)	Call Reason	
starttime	INT(4)	Call Start Time	
duration	INT(4)	Total Call Duration	
startwait	INT(4)	Wait Start Time	
durationwait	INT(4)	Total Wait Time	
startconnect	INT(4)	Connect Start Time	
durationconnect	INT(4)	Total Connected Time	
startwrap	INT(4)	Wrap Start Time	
durationwrap	INT(4)	Total Wrap Time	
transferredfrom	VARCHAR(40)	Callid of transferred from field	

Payload Table Schema definition

The payload table provides persistence for application specific data relating to calls. This can be custom defined IVR data / call variables, Agent Manager API

Field Name	Data Type	Description	Default
urn	INT(8)	Unique reference number – Auto increment	Next URN
callid	VARCHAR(40)	Unique call ID to which this payload relates	
data	VARCHAR(2048)	XML encoded payload, application defined	

ADR Table Schema definition

The Publisher.DB - ADR table represents the primary persistence layer for all ADR records / events in the system. ADR records relate to agent activity, the events of which are generated from agent manager activity.

Field Name	Data Type	Description	Default
urn	INT(8)	Unique reference number – Auto increment	Next URN
starttime	INT(4)	Event Start time as POSIX time	N/A
username	VARCHAR(40)	ContactQ User Name	N/A
firstname	VARCHAR(40)	User's configured first name concatenated into a string, separated with a space.	N/A
lastname	VARCHAR(40)	User's configured last name concatenated into a string, separated with a space.	N/A
activity	VARCHAR(3)	Activity Identifier LI = Login LO = Logout AV = Available UA = Unavailable IC = Inbound Call OC = Outbound Call AT = Alert Time (Inbound) DT = Dial time WI = Wrap-up Inbound WO = Wrap-up Outbound	N/A
reason	VARCHAR(80)	The reason describing this state, e.g. Presence State, Disposition, ContactID etc.	NULL
sessionid	VARCHAR(40)	The session identifier associated with the agent for which this event is pertinent	N/A
callid	VARCHAR(40)	An associated callID for this state. NULL if not applicable.	N/A
mediaid	VARCHAR(40)	An associated MediaID for this state. NULL if not applicable. This media tag can associate a call originator with a media type such as Voice, Video, Chat &c.	

API (XML RPC)

ContactQ user and administrative functionality can be accessed remotely through an XML-RPC API. XML-RPC is a remote procedure call protocol which uses XML to encode its calls and HTTP as a transport mechanism. Underlying the protocol is an HTTP 1.1 compliant web server which hosts RPC requests and associative file transfer requests. The API contains a registry of method interfaces enabling the programmatic remote control of system resources and configuration using a standard wire protocol. The API is self documenting when an introspection interface is enabled. Interfaces supported in version 1 of the API include user call control, system presence management, and a comprehensive set of call centre functions.

```
POST /RPC2 HTTP/1.1
User-Agent: CQ/1.1
Host: demo.contactq.org
Content-Type: text/xml
Content-length: 181
```

```
<?XML VERSION="1.0"?>
<methodCall>
  <methodName>ContactQ.API.Echo</methodName>
  <params>
    <param>
      <value><string>Hello ContactQ</string></value>
    </param>
  </params>
</methodCall>
```

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Sun, 1 Feb 2010 06:00:10 GMT
Server: ContactQ 0.1 Server.Demo/1.1
```

```
<?XML VERSION="1.0"?>
<methodResponse>
  <params>
    <param>
      <value><string>Hello ContactQ</string></value>
    </param>
  </params>
</methodResponse>
```

ContactQ Protocol

The ContactQ system uses a single protocol for all communications between its applications and gateways. The protocol is text based and is similar to both HTTP and SIP. Each message comprises of a number of MIME headers followed by one or more body entities that may include file attachments. Each ContactQ message is passed as an XML document within a body entity.

- **Mime Headers**

ContactQ uses the following headers:

To: registrar@localhost:4080

From: acd@localhost:4081

Content-Type: application/xml

Content-Transfer-Encoding: binary

Content-Length: 0

Both multipart messages and file attachments are supported.

- **Transport**

The transport layer uses TCP/IP. The ContactQ protocol does not require that connections are persistent as connections are initiated on-demand. However, the normal behaviour is for connections to be established during startup and to persist until one party exits.

- **Message Routing**

ContactQ messaging is a peer-to-peer system i.e. each end-point is both a client and a server and messages may be routed directly between them. When running on a single host this means that each end-point listens on a different port. By default ports starting at 4080 are used; these are marked 'unused' in the IANA port assignments document. Currently ports 4080-4090 have been utilised in the default configuration of the ContactQ application appliance server instance, however all ports are configurable for other deployments.

LDAP

The Lightweight Directory Access Protocol (LDAP) is an Internet protocol for accessing distributed directory services that act in accordance with X.500 data and service models. The ContactQ configuration and user authentication interface supports the enumeration of directory resources and synchronisation with remote directories to support a common directory within an organisation.

SMTP

An SMTP mail gateway is a module of the ContactQ platform which is configurable at a domain level enabling registered applications of the platform to send mail using simple or authenticated SMTP sessions. SMTP integration enables the ContactQ Mail Gateway process to open a TCP connection to a server SMTP process on a remote host and attempt to send mail across the connection. The server SMTP listens for a TCP connection on a well-known port (25), and the user SMTP process initiates a connection on that port. When the TCP connection is successful, the local and remote processes engage in a simple request/response discourse, defined by the SMTP protocol, in which the originating process transmits the mail addresses of the originator and the recipient(s) for a message. The destination SMTP server process accepts these mail addresses; the user process transmits the message.

Service Gateway HTTP(S)

A request response service gateway implements dynamically configurable services into the platform including HTTP(S). Registered applications are able to make use of remote services using this interface. Typical use cases would include integrating payment gateways, remote data access management services or remote speech services into the IVR module.

SMXML

State Machine XML (SMXML) is an event-based hierarchical state machine language designed for call control in ContactQ applications. It is similar to State Chart XML (SCXML) which is a candidate for the control language within VoiceXML 3.0 (currently under development by the W3C Voice Browser working group), although with a simplified feature set. Complex customisation of the ContactQ application platform is achievable by scripting dialects of supporting applications including but not exhaustively the ACD, IVR, ACD and AgentManager application server modules.

SNMP

Simple Network Management Protocol (SNMP) is a UDP-based network protocol which is used mostly in network management systems to monitor network-attached devices for conditions that merit awareness. SNMP is a component of the Internet Protocol Suite as defined by the Internet Engineering Task Force (IETF). It comprises a set of standards for network management, including an application layer protocol, a database schema, and a set of data objects. SNMP exposes management data in the form of variables on the managed systems, which describe the system configuration. These variables can then be queried (and sometimes set) by managing applications.

PEN

SMI Network Management Private Enterprise Codes:

Prefix: iso.org.dod.internet.private.enterprise (1.3.6.1.4.1)

Braxtel Communications Inc has been assigned the Private Enterprise Number (PEN) 29301 by [IANA](#).

BRAXTEL-MIB.txt

```
/usr/share/snmp/mibs/BRAXTEL-MIB.txt
BRAXTEL-MIB DEFINITIONS ::= BEGIN

IMPORTS
    enterprises
        FROM SNMPv2-SMI;

braxtel MODULE-IDENTITY
    LAST-UPDATED      "200710080900Z"
    ORGANIZATION      "Braxtel Communications Inc."
    CONTACT-INFO
        "John Tarlton
        Email: jtarlton@braxtel.com"
```

```
DESCRIPTION
    ""
    ::= { enterprises 29301 }

END
```

IP Port

UDP Port 161

Management Information Bases (MIBs)

SNMP uses Management Information Bases (MIBs) to specify the management data of a device subsystem,.

Linux stores MIB files in the directory: /usr/share/snmp/mibs

SNMP Agents

ContactQ processes operate as AgentX Sub-Agents that connect to the local Master Agent (snmpd).