Description of the Remote API

DevTech Services, LLC

Contact DevTech at http://www.devtechservices.com or info@devtechservices.com.

The Remote API provides a method for extending and automating input and output to and from BMC Software's Magic Service Desk. It allows for command line interfacing, monitored actions, and direct connections between Magic and external data stores, services, and applications.

Overview of the Remote API

The core component of the Remote API is the server application. The server application resides on a Magic web/application server. Its primary function is to respond to requests for actions from external sources and return data based on those request.

The server consists of two components. The listener runs as a service and waits for communications from external clients, either local or remote, and responds as needed. The actor is an ActiveX component called by Magic to initiate outbound communication.

Working in conjunction with the server application are one or more client applications. Depending on the specific requirements of the function being performed, the client may reside on the Magic web/application server or a remote host. Generic clients for Windows and Unix platforms are available. This client is compatible with many major network management packages, such as the SMARTS InCharge suite (http://www.smarts.com.) Other pre-configured clients are available for a variety of applications and services. Custom client can also be created by request.

Requirements

The Remote API Server components share the same hardware and software requirements as the Magic Service Desk application server. The Remote API Server must be installed on a Magic Web/Application server.

If the Remote API Server Sender component is to be used, the Magic MBA Option Pack is required.

The Remote API Java Client requires Java Runtime Environment version 1.3 or better and is platform independent.
The Remote API Win32 client requires:

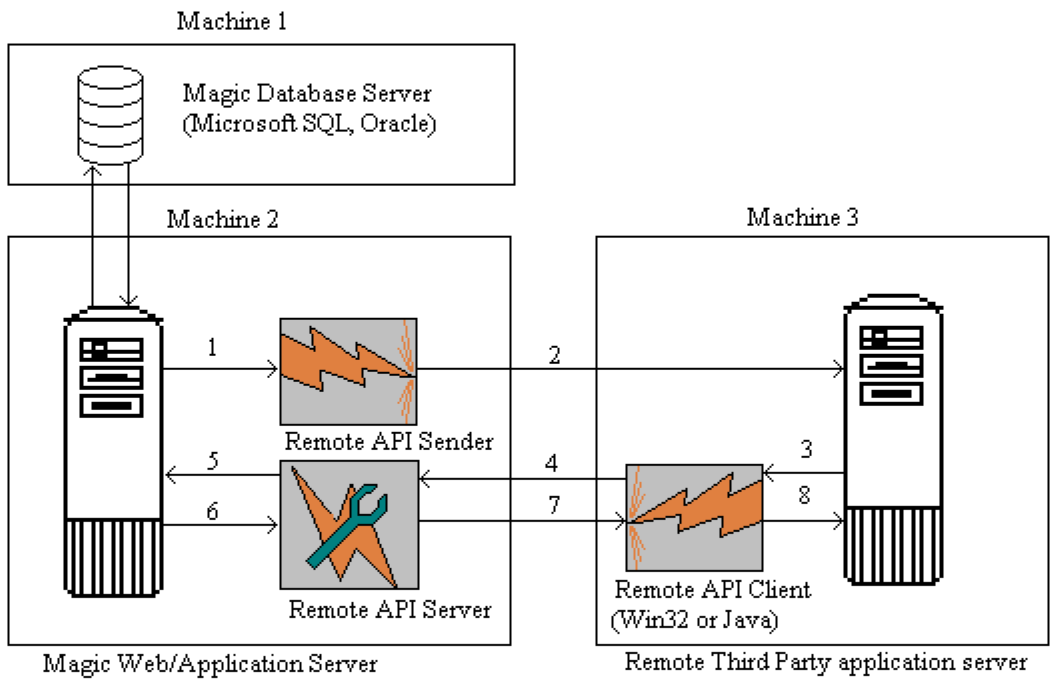
Windows NT 4 with Service Pack 6 or
Windows 2000 Server with Service Pack 3
Microsoft Data Access Components (MDAC) version 2.7 or higher

Hardware requirements for the Win32 client are the same as for the operating systems.

Sample Topology

A simple topology for the Remote API consists of a Magic system with the Remote API server, a third party application capable of accessing a command line interface, and a Remote API client. Individual communication events are initiated from either the client or the server side. Figure 1 shows a simplified topology diagram where the Remote API is used to communicate with a generic client on a remote application server. The numbers represents logical steps in the process of communicating between the two servers.

Figure 1



Magic Initiated actions

Step 1: An event occurs in Magic that triggers action by the Remote API Sender
Example: A Magic Incident is created with a field named “Export” set to 1. Magic Business Automation creates a new record in an action module. The record creation

triggers the Remote API Sender to send data to the specified remote host. See How the Remote API Manipulates Data for information on how data is controlled and transformed between applications.

Step 2: The Remote API Sender sends information from the triggering record to the remote application. Depending on the remote application's requirements, the data is sent via ODBC (a SQL query,) network protocol (such as an SNMP SET command,) or RPC (usually by a remote client native to the remote application.)

Step 3: The remote host receives the data from the Remote API Sender. Events may or may not be triggered at the remote application. If an event is triggered, the remote application may be set to call the generic client, usually by command line interface.

Step 4: If the generic Remote API Client is invoked, it sends data to the Remote API Server depending on conditions set by the remote application. Data from the remote application is included in the communication, along with instructions on what action to perform, and what data to retrieve.

Step 5: The Remote API Server receives the data from the Client. The data contains information regarding the appropriate action the server should take (Create, Retrieve, Update, or Delete,) the Magic module to be affected (Incident, Work Orders, or any other module,) if necessary, the unique identifier for the record (Incident #, for example,) data to store in the new or updated record by field and value, and for fields for which data should be retrieved.

Step 6: The Server performs the specific action in Magic, then retrieves information for the new, updated, or requested record. See How the Remote API Manipulates Data for information on how data is controlled and transformed between applications.

Step 7: The Remote API Server responds to the Remote API Client with the data the Client requested.

Step 8: The Remote API Client updates the remote application with the information.

Remotely Initiated Actions

In some cases, actions are initiated by the remote application. In these cases, the flow is the same as above, but starts with Step 3. This is an example of the flow initiated by a network event being detected by network management software, starting at Step 3 in Diagram 1.

Step 3: A web server named WEBSERVER1 loses contact with the management server, causing the management package to create an event showing that the server is down. The management software is scripted to call the Remote API Client's command line interface when certain events occur, and this is such an event.

Step 4: The Client contacts the Remote API Server and forwards the action request, data for Magic, and column values to return to the application. The Action is to "Create" an Incident where Client ID = 'WEBSERVER1', the Incident Description is "Server is down," Impact is 1, Severity is 1, Status ID of "OPEN", and the Subject ID is "Server."

Step 5: The Remote API Server then creates the Incident and relevant parent records.

a. It checks to see to see if a Client with Client ID "WEBSERVER1" exists, and doesn't find it.
b. It then checks to see if it should translate "WEBSERVER1" into another value when storing that as a Client ID, but does not find a translation.
c. It creates the new Client record by first checking to see if there are any default values for Client records, and then checking if there are any other fields then Remote Client has sent for the Client record that appear on the Incident screen (First Name, Last Name, etc.) If any other values are sent from the Client, they overwrite any default values.
d. The Remote API Server goes through sub-step c for each field sent from the Remote API Client, and then checks for any default values for fields not sent.
e. After creating all necessary parent records, it creates the Incident itself.

See How the Remote API Manipulates Data for information on how data is controlled and transformed between applications.

Step 6: The Remote API Server retrieves the requested column values.

Step 7: The Remote API Server sends the results back to the Remote API Client.

Step 8: The Remote API Client updates the remote application with the data from Magic.

In either case, any Magic Business Logic concerned with the creation of records by the Remote API will be triggered into the Job Queue.

How the Remote API Manipulates Data

In many cases, when the Remote API Client requests a record creation, it will supply all relevant values to the Remote API Server. The Server will then create the record as is with the values given.

To provide extensible logic to the Remote API's capabilities, several methods are available for monitoring and changing data.

Defaulting Values

For each field in each module, a default value can be set. When the Remote API attempts to create a record for a module, it checks for any default values and assigns them first. If the server then finds a value supplied by the Client, it discards the default value and uses the value provided by the client.

Data Translation

Direct translation

Once the server has identified a value for a field, it checks to see if it should be translated into another value. This allows data of one logical type to be converted into another. For example, the client may request that an Incident be created with a "Login ID Assigned To" value of "SERVER". The Remote API Server will check to see if there are any translation pairs for "Login ID Assigned To" in the "Incident" view. The translation would be set so that the value "SERVER" for "Login ID Assigned To" should translate to "BROWNK". Therefore, the Remote API Server would create the Incident and assign it to Login ID BROWNK.

For each translation for a field, one translation pair should exist. Multiple incoming values can be set to translate to the same value for inserting or updating into Magic.

Scripting

Another method of translation is through the use of VBScript scripting. The Remote API can refer to user-defined VBScript functions to modify values, lookup other values, and to perform other actions such as calling ActiveX objects on the server.

Creating Parent Records

The Remote API Server receives data in discrete values for each field for the module specified by the Remote API Client. For example, the Remote API Client can request that a new Incident be created with a Client ID of "TESTCLIENT." If the Client record exists for "TESTCLIENT," then the Remote API Server will use the existing record. However, if no record is found to fill the foreign key appropriately, the Remote API can be set to create the new record. If so, then the Remote API goes through the same process of defaulting and data translation as above. Also, if more than one field value is given for the parent record (Client ID and First Name and Last Name for a Client record, for example,) all are used in creating the new record.