CCS LANGUAGE REFERENCE
FOR
CREACODE SIP APPLICATION SERVER®

Revision History

Version	Date	Author	Comments
1.0.6.2	2004-12-23	Arda Tekin	First draft
1.0.7.0	2005-05-03		Voicemail functions added
1.0.8.0	2005-09-08		Database ODBC functions added
1.1.0.4	2006-09-12		_INCOMING_DOMAIN_ script constant added
1.1.2.4	2009-04-06		GetRequestHeaderByName function added

Table of Contents

CCS Language Reference For Creacode SIP Application Server®

DATA MANIPULATION

Data Types

The scripting language supports 4 types of data

- integer
  Exp. value: 2004
- string
  Exp. value: "033244455"
- bool
  Exp. value: 0 or 1
- double
  Exp. value: 1550,35

Data can be used in the script files in the followings forms:

- User local variables
- User global variables
- System variables

Variables

Variables are interpreted according to the place they are defined. Variables can be defined in three different scope:

- Application global variable: Defined on the top of main.ccs file and can be accessed by all other script files associated that application.
- Script global variable: Defined on top of any script file. These kind of global variables are accessible only for the script file they are defined in.
- Local variable: Defined inside an event or a function block and only valid and accessible in that block.

OPERATORS

Aritmetic Operators

Variables and data values can be processed within aritmethical forms.

Exp:

```
int A;
int B;
int C;
string strA;
string strB;
string strC;

A = 6;
B = 3;
strA = "01";
strB = "3551234567"
C = A + B;        (Add)
C = A – B;        (Substract)
C = A / B;        (Divide)
C = A * B;        (Multiply)
C = A % B;        (Mod)
strC = strA+strB  (Concat 2 string)
```

Comparison Operators

Logical expressions are specially used in conditional statements such as "if, for, while".
Available logical expressions:

```
==      : Equal (For integer and string)
>       : Greater then
<       : Less then
>=      : Greater then or equal
<=      : Less then or equal
!=      : Not equal
```

Logical Operators

```
!       : Not
&&      : And
||      : Or
```

SELECTION STATEMENTS

if Statements

Used to create for "if ", "else if" blocks.

Example

```
int max_call_duration = 1800;
if (max_call_duration < 2000  ) {
        ...
}
```

```
string strDigits;
if (strDigits == "0326665444") {
        ConnectToCallCenter();
}
else if (strDigits == "0326665441") {
        ConnectToMarketing();
}
```

ITERATION STATEMENTS

for Loops

Used to create "for" loops.

```
for (expression1; expression2; expression3) {
        statement1;
        statement2;
        ...
}
```

Example

```
int i = 0;
for (i = 0; i < 3; i++) {
        g_strDigits = GetDigits(_LEG_A_, USER_DIGIT_LEN, "pincode_enter");
        LOG ("g_strDigits:" + g_strDigits);
}
```

Remarks

It is possible to define expression1 in parenthesis like in C++.
Exp: for (int i = 0; i < 3; i++)...

while Loops

Used to create "while" loops.

```
while (expression) {
        statement1;
        statement2;
        ...
}
```

break Keyword

"break" keyword is used to interrupt the execution of loop.

eave out from the for and while loop.

Example

```
while (nLen < g_UserCodeLen) {
        ...
        if (nLen == g_UserDigitLen)
                break; // Suppose to get out the loop
}
```

CCS API Reference

Signalling Functions

void AcceptCall(int nMsgCode);

Used to send RINGING(non-SDP) message to **A** side.

Parameters

nMsgCode (optional)

RINGING message code in SIP. Defult is 180.

void AcceptCallWithMedia(int nMsgCode);

Used to send RINGING(with-SDP) message to **A** side.

Parameters

nMsgCode (optional)

RINGING message code in SIP. Defult is 180.

void AnswerCall();

Used to send that IVR has answered the call. (IVR Sends 200 OK in SIP)

void StartCall(nLegSrc, nLegDst, strCalledNumber);

Used to make call to the called number. IVR sends INVITE in SIP.

Parameters

nLegSrc

Calling party.

nLegDst

Called party.

strCalledNumber

The telephone number of the called part.

Remarks

Possible values for nLegSrc and nLegDst are:

_LEG_A_	Calling part
_LEG_B_	Called part
_LEG_IVR_	IVR
_LEG_NA_	Unknown part

Example

```
StartCall(_LEG_A_, _LEG_B_, "0324234567");
```

void RejectCall(int nCause);

IVR sends Call Reject message with the specified cause code.

Parameters

nCause (optional)

Cause code of reject message. Default is 488.

void JoinLegs(int nLegSrc, nLegDst);

Provides to modify the media information of a part. (RE-INVITE in SIP)

Parameters

nLegSrc

The part which sends its new media information

nLegDst

The part which receives the new media information

Remarks

nLegSrc possible values are _LEG_A_, _LEG_B_. _LEG_IVR_

nLegDst possible values are _LEG_A_, _LEG_B_

JoinJegs(_LEG_A_, _LEG_B_): IVR sends new SDP of LEG_A to LEG_B.
JoinJegs(_LEG_B_, _LEG_A_): IVR sends new SDP of LEG_B to LEG_A.
JoinJegs(_LEG_IVR_, _LEG_A_): IVR sends IVR SDP to LEG_A. (Binds IVR to LEG_A)
JoinJegs(_LEG_IVR_, _LEG_B_): IVR sends IVR SDP to LEG_B. (Binds IVR to LEG_B)

Example

```
// Play "goodbye" announce when maximum call duration has expired.
// First, close the B party.
EndCall(_LEG_B_) ;
JoinLegs(_LEG_IVR_, _LEG_A_);
PlayAudio(_LEG_A_, "goodbye");
EndCall(_LEG_A_);
```

void SendDigit(int nLeg, int nDigit, int nDuration);

Sends a digit to the specified LEG in a SIP INFO packet.

Parameters

nLeg
: The LEG which the SIP INFO packet to be sent to.

nDigit
: Digit to be sent. 10=*, 11=#

nDuration
: Digit duration in millisecond.

void RedirectCall(int nDest);

Redirects received 200 OK SIP packet to the opposite side.

Parameters

nDest
: The name of destination party.

Remarks

Possible nDest values are _LEG_A_, _LEG_B_.

void EndCall(nLeg);

IVR sends SIP BYE message to that side.

Parameters

nLeg
: The LEG to be closed.

Remarks

EndCall(_LEG_IVR_) is equal to EndCall(_LEG_A_) and then EndCall(_LEG_B_).

void ReleaseSession();

Releases all open recources related to session and then closes the session.

Remarks

ReleaseSession function must be called to close the active session. Generally used in CallEnd event block.

Voice Functions

void OpenAudioChannel(int nLeg);

Opens an RTP session to specified leg.

Parameters

nLeg

Opens an audio channel towards that side.

Example

OpenAudioChannel(_LEG_A_);

void CloseAudioChannel(int nLeg);

Closes the active RTP session for the specified leg.

Parameters

nLeg

The LEG that its RTP session to be closed.

Example

CloseAudioChannel(_LEG_A_);

void PlayAudio(int nLeg, string strAudioFileName, string strSessionLanguage);

Plays audio to the specified leg.

Parameters

nLeg

NicIVR plays audio to that LEG.

strAudioFileName

The name of the file which contains the related audio.

strSessionLanguage(optional)

Session language string.

Example

PlayAudio(_LEG_A_, "getpincode.wav", "english");

void PlayBackgroundAudio(int nLeg, string strAudioFileName, bool fCyclic, int dCyclicWaitTime);

Plays audio in background to the specified leg.

Parameters

nLeg

NicIVR plays audio to that LEG.

strAudioFileName

The name of the file which contains the related audio.

fCyclic

If this parameter is TRUE, plays the annouce in cyclic mode.

dCyclicWaitTime

Wait period for cyclic playing in seconds.

void StopBackgroundAudio(int nLeg);

Stops the background audio.

Parameters

nLeg

Stops the playing audio for that side.

string GetDigits(int nLeg, int nNumberOfDigits, string strAudioFileName, string strSessionLanguage);

Gets DTMF signals from the specified leg. Returns the captured digits.

Parameters

nLeg

NicIVR captures digits from that side.

nNumberOfDigits

Maximum number of digits to be captured.

strAudioFileName
The name of the file which contains the related annouce.
strSessionLanguage(optional)
If this parameter is TRUE, plays the annouce in cyclic mode.

Remarks
GetDigits sets the _STATUS_ internal variable. Possible values in this system variable are
0: OK
1: Digit Internal Error
2: Digit Timeout
3: Session Not Created
4: Voice Data Not Found
5: Stop Message

Example
string capturedDigits = GetDigits(_LEG_A_, 5, "getpincode.wav", "english");

void PlayCredit(int nLeg, int nBalance, string strBalanceAudioFile);

Plays amount of credit to the specified leg.
Parameters
nLeg
NicIVR plays credit to that side.
nBalance
Credit balance value.
strBalanceAudioFile
File name of the balance announce.
Example
```
//First play integer part of the credit
PlayCredit(_LEG_A_, 10, "dollar");
//Then play the decimal part
PlayCredit(_LEG_A_, 7, "cent");
```
Remarks
Following audio files for numbers should be placed under voice codec directory.
Say_0, Say_1, Say_2, Say_3 .... Say_99
Say_100, Say_200, Say_300.... Say_900, Say_1000

PlayTime(int nLeg, int nDuration);

Announces the time in hour-minute-second format to the specified leg.
Parameters
nLeg
NicIVR plays the time to that side.
nDuration
Duration in seconds.
Remarks
Audio files named "hours", "minutes", "seconds" must also be placed in audio directory.
Following voice files for numbers should be placed under voice codec directory.
Say_0, Say_1, Say_2, Say_3 .... Say_99
Say_100, Say_200, Say_300.... Say_900, Say_1000

Utility Functions

void ReturnScript(string strScriptFileName, string strEventName);

Starts to process the specified file.

Parameters

strScriptFileName

Name of script file to be processed.

strEventName

Name of the event in the specified file.

Remarks

If the event specified in the second parameter is in the script file, this event is processed immediately.

Example

ReturnScript("GetUserID.ccs", "AuthSuccess");

int StartTimer(int nDuration, string strEvent, bool fOneShot);

Starts a timer. When the timer expires, calls the event specified in strEventName parameter. Returns created TimerID value.

Parameters

nDuration

Duration in seconds.

strEventName

Event to be processed in the current or global scope when the timer expired.

fOneShot

If TRUE, strEvent function is called only once.

string GetTime();

Returns seconds since 00:00:00 UTC, January 1, 1970 in string format.

string FormatTime(string strTime, [string strFormat]);

Creates formatted date/time value.

Parameters

strTime

Time in seconds (Return value of GetTime() function)

strFormat

Date/time formatting string. If this parameter is empty or "UTC", return value is in UTC format. (Exp: 17:30:35.150 UTC Sat Jun 04 2005)

%a Abbreviated weekday name
%A Full weekday name
%b Abbreviated month name
%B Full month name
%c Date and time representation appropriate for locale
%d Day of month as decimal number (01 – 31)
%H Hour in 24-hour format (00 – 23)
%I Hour in 12-hour format (01 – 12)
%j Day of year as decimal number (001 – 366)
%m Month as decimal number (01 – 12)
%M Minute as decimal number (00 – 59)
%p Current locale's A.M./P.M. indicator for 12-hour clock
%S Second as decimal number (00 – 59)
%U Week of year as decimal number, with Sunday as first day of week (00 – 53)
%w Weekday as decimal number (0 – 6; Sunday is 0)
%W Week of year as decimal number, with Monday as first day of week (00 – 53)
%x Date representation for current locale
%X Time representation for current locale
%y Year without century, as decimal number (00 – 99)
%Y Year with century, as decimal number
%z,**%Z** Either the time-zone name or time zone abbreviation, depending on registry settings; no characters if time zone is unknown
%% Percent sign

int TimeDiff(string strTime1, string strTime2);

Substracts strTime2 from strTime1. Result is seconds in string format.

Parameters

strTime1
: End time

strTime2
: Start time

int GetDuration();

Returns the duration in seconds since the call established between LEG_A and LEG_B. GetDuration function returns 0 when the telnet command "kill [session_id] notbill" is used.

void Sleep(int nNumberOfSeconds);

Stops the CCS execution until specified duration expires.

Parameters

nNumberOfSeconds
: Wait time in seconds.

string GetStatusMsg();

Returns the last occured internal error message.

void LOG(string strUserLogMessage);

Writes string message into script.log file for script logging purposes.

Parameters

strUserLogMessage
: Message string which will write into script.log file.

Example

```
LOG("Script is in GetPinNumber function");
```

void SendMail(string strFrom, string strTo, string strCC, string strSubject, string strBody,string strMailServer);

Sends e-mail using the paremeters given.

Parameters

strFrom
: Mail From

strTo
: Mail To

strCC
: Carbon copy

strSubject
: Mail Subject

strBody
: Mail Body

strMailServer
: Mail server IP adress or domain.

Remarks

SendMail function leave an e-mail to NicIVR e-mail sender queue. Hence this function doesn't block script execution i.e. runs asyncronously.

string GetRequestHeaderByName(string strHeaderName);

Returns value of requested header name.

Parameters

strHeaderName
: Name of header in incoming SIP message

Example

```
string genHeader = GetRequestHeaderByName("max-forwards");
// Would probably return '70'
```

HTTP Messaging

void AddXMLParam(string strName, string strValue);

Adds a request line into Request XML message.

Parameters

strName
: name attribute of the element.

strValue
: value attribute of the element.

Remarks

Use this function to add name-value pairs before posting the Request XML message.

Example

```
AddXMLParam("Pincode", "1123487882");
AddXMLParam("Password", "4868");

// 2 elements line will be added the Request XML message.
<?xml version="1.0"?>
<Request>
        <Element name="Pincode" value="1123487882"/>
        <Element name="Password" value="4868"/>
</Request>
```

int PostHttpXML(string strServerIP, string strPageName);

Posts the final Request XML message to the specified page. Returns TRUE if the posting is successful.

Parameters

strServerIP
: IP of the application server in dotted format.

strPageName
: Page name, starts with "/".

Remarks

If PostHttpXML does not return TRUE, check the post failure reason with GetLastRuntimeErrorMessage() function.

Example

```
PostHttpXML("65.200.216.25", "/PinValidate.aspx");
```

string GetXMLParamValue(string strName);

Gets the value of the specified parameter from Response XML.

Parameters

strName
: Name parameter of the response element.

Example

```
//Response XML received from webserver
<?xml version="1.0"?>
<Response>
        <Element name="IsValidated" value="1"/>
        <Element name="Balance" value="100"/>
</Response>

string strValidated = GetXMLParamValue("IsValidated");
string strBalance = GetXMLParamValue("Balance");
//strValidated is "1" and strBalance is "100"
```

void ResetXML();

Clears all the Request XML content.

Remarks

Use this function after getting the Response XML parameters

RADIUS Messaging

void AddRadiusAttr(int nRadiusID, string strRadiusValue);

Adds a Radius attribute value into request.

Parameters

nRadiusID
Radius attribute ID.

strRadiusValue
value of the attribute.

Example

AddRadiusAttr(1, g_strUserID);

void AddRadiusVSAAttr(int nVSAID, string strVSAValue, int nVendorID);

Adds a Vendor Specific Attribute value into request.

Parameters

nVSAID
VSA ID.

strRadiusValue
value of the VSA.

nVendorID
Vendor ID.

int SendRadiusRequest(int nRequestType, string strServerIP, int nServerPort);

Sends the added radius attributes to the specified radius server.

Parameters

int nRequestType
Radius request type.

strServerIP
Radius server IP.

strValueType
Radius server port.

Remarks

nRequestType can be following values
-ACCESS_REQUEST : Equals to 1
-ACCOUNTING_REQUEST : Equals to 4

Return Values
-Returns 0 : Access Accept received
-Returns 1 : Access Reject received
-Returns 2 : Techical problem occured. Check data post failure reason with GetLastRuntimeErrorMessage() function.

string GetRadiusAttrValue(int nRadiusID);

Returns specified radius attribute value in the response message.

Parameters

nRadiusID
Radius attribute ID.

string GetRadiusVSAAttrValue(int nVSAID);

Returns the specified VSA value in the response message.

Parameters

nVSAID
VSA ID.

void ResetRadius();

Resets the radius session.

Remarks

Clears the request and response content.

VoiceMail Functions

string RecordVoice(int nLeg, int nDuration, string strInterruptDigit);

Records voice from a leg. Returns the full path of the recorded file.

Parameters

int nLeg
: NicIVR records voice from this LEG

int nDuration
: Record duration in seconds

string strInterruptDigit
: Record intertupts when pressing this digit

void RecordnMailVoice(int nLeg, int nDuration, string strInterruptDigit, string strMailAddress);

Records voice and sends recorded file to specified e-mail address. This function runs asyncronously.

Parameters

int nLeg
: NicIVR records voice from this LEG

int nDuration
: Record duration in seconds.

string strInterruptDigit
: Record intertupts when pressing this digit.

string strMailAddress
: E-mail address for the record to be sent.

Database Connectivity Functions

int ODBCConnect(string strConnectionString);

Opens an ODBC connection to database.

Parameters

string strConnectionString

Specifies an ODBC connection string. This includes the data source name, user ID and password. For example, "DSN=SQLServer_Source;UID=SA;PWD=abc123" is a possible connection string.

Remarks

Returns unique Connection ID which will be used by other ODBC functions.

int ODBCDisconnect(int nConnectionID);

Closes the ODBC connection. Returns nonzero if the function success. If fails returns 0.

Parameters

int nConnectionID

Unique connectionID

int ODBCExecute(int nConnectionID, string strQuery);

Executes INSERT, UPDATE or DELETE query. Returns nonzero if the function success. If fails returns 0.

Parameters

int nConnectionID

Unique connectionID

string strQuery

Execute query statement.

int ODBCExecuteSelect(int nConnectionID, string strQuery);

Executes a SELECT query. Returns nonzero if the function success. If fails returns 0.

Parameters

int nConnectionID

Unique connectionID

string strQuery

Execute query statement.

int ODBCIsEOF(int nConnectionID);

Nonzero if the recordset contains no records or if you have scrolled beyond the last record; otherwise 0.

Parameters

int nConnectionID

Unique connectionID

void ODBCMoveNext(int nConnectionID);

Makes the first record in the next rowset the current record.

Parameters

int nConnectionID

Unique connectionID

int ODBCGetIntField(int nConnectionID, string strFieldName);

Returns the integer value in the specified field.

Parameters

int nConnectionID

Unique connectionID

string strFieldName

Column name of the current record.

double ODBCGetDobuleField(int nConnectionID, string strFieldName);

Returns the double value in the specified field.

Parameters
int nConnectionID
Unique connectionID
string strFieldName
Column name of the current record.

string ODBCGetStringField(int nConnectionID, string strFieldName);

Returns string value in the specified field.

Parameters
int nConnectionID
Unique connectionID
string strFieldName
Column name of the current record.

String Manupulation Functions

int strlen(string strData);

Returns the length of the string.

Parameters

strData

String data

int str2i(string strData);

Converts a string to integer.

Parameters

strData

String which contains numbers.

Remarks

If the param does not contain numbers, the function returns -1.

string i2str(int nData);

Converts an integer value to string.

Parameters

nData

Number

int strfind(string strSrc, string strSub, [int nStartIndex]);

Returns the first occurence index of search-string in the source string.

Parameters

strSrc

Source string

strFindData

Search string

startIndex

Zero based index to start the search. (If not specified nStartIndex=0)

Remarks

If the search string can not be found in the source string, function returns -1.

string strleft(string strSrc, int nCount);

Returns specified lenght of characher from the left of the source string.

Parameters

strSrc

Source string

nCount

Number or characters to be extracted.

string strright(string strSrc, int nCount);

Returns specified lenght of characher from the right of the source string.

Parameters

strSrc

Source string

nCount

Number or charachers to be extracted.

string strmid(string strSrc, int nStart, int nCount);

Returns a string containing a specified number of characters from a string.

Parameters

strSrc

String expression from which characters are returned.

nStart

Character position in *strSrc* at which the part to be taken starts

nCount

Number of characters to return.

Mathematical Functions

int neg(int nNumber);

Returns negative value of the number.

Parameters

nNumber

Integer/double value.

double neg(double nNumber);

Same as ***neg*** function.

int rand(int nStart, int nEnd);

Returns random number.

Parameters

nStart

Random number lower limit.

nEnd

Random number upper limit.

string randHex(int nDigitCount);

Returns hexadecimal random number in string format.

Parameters

nDigitCount

Specifies number of digit to be generated.

Internal System Variables

The system has some internal constants.

TRUE
Specially used in logical expressions. Means the same as integer 1.
Type : integer, bool
Access : GET

FALSE
Means the same as integer 0.
Type : integer, bool
Access : GET

_IVR_IP_
Dotted format system ip.
Type : int
Access : GET

_SESSION_ID_
Current session id.
Type : string
Access : GET

_LEG_A_
Calling part.
Type : integer
Access : GET

_LEG_B_
Called part.
Type : integer
Access : GET

_LEG_IVR_
System side
Type : integer
Access : GET

_LEG_NA_
Unknown part.
Type : integer
Access : GET

_CLOSING_LEG_
Closing(Hang-up) part.
Type : integer
Access : GET

_CALLING_NR_
Calling number.
Type : string
Access : GET

_CALLED_NR_
Called number.
Type : string
Access : GET

LANGUAGE
Active session language. (string)
Type : string
Access : GET/SET

RESPONSE
Last SIP response code received by the system.

Type : integer
Access : GET

DIGIT
Stores incoming digit.
Type : int
Access : GET

_DIGIT_DURATION_
Stores incoming digit duration in milliseconds.
Type : int
Access : GET

_INCOMING_DOMAIN_
Stores the Domain information in the "To" field of first received INVITE message.
Type : int
Access : GET

CCS Events

Signalling Events

EVENT NewCall()

Processed when the new call is received to IVR.

EVENT CallRinging()

Processed when the called part sends RINGING message without SDP.

EVENT CallRingingWithMedia()

Processed when the called part sends RINGING message with SDP.

EVENT CallReject()

Processed when the called part state is busy, calling but not answer or the called number is not available.

EVENT CallAnswered()

Processed when the destination answers the incoming call.
(200OK Received from B)

EVENT CallEstablished()

Processed when the call is established between origination and destination.
(200OK Received for Re-INVITE to A)

EVENT CallActive()

Processed when the call is established between origination and destination.
(ACK Received from A for 200OK)

EVENT CallEnd()

Processed when one of the leg ends the call.

EVENT OnDigit()

Processed when a digit is received in SIP INFO packet. _DIGIT_ and _DIGIT_DURATION_ global variables gives information about incoming digit.

EVENT OnCallChangedByLegA()

This event is interpreted when LEG_A part sends a new media information to LEG_IVR. (re-INVITE from LEG_A)

EVENT OnCallChangedByLegB()

This event is interpreted when LEG_B part sends a new media information to LEG_IVR. (re-INVITE from LEG_B)

Scripting Events

EVENT < [TimerFunctionName] >()

If a timer that is starting with StartTimer event expires, the event specified in the "EventName" parameter in the StartTimer function is processed.

Example

```
{
        .....
        //600 seconds later the event named "TimerFunction" will be called
        StartTimer(600, "TimerFunction", TRUE);
        .....
}

EVENT TimerFunction()
{
        PlayAudio(_LEG_A_, "No_balance");
        PlayAudio(_LEG_A_, "Goodbye");
        ReleaseSession();
}
```

EVENT < [UserEventName] >()

Custom events are called using RunScript(<[ScriptName]>, <[CustomEventName]>) function.

Example

```
//In the NewSession.ccs script file
{
        ....
        RunScript("GetPinNumber.ccs", "UserEvent");
}

//In the GetPinNumber.ccs script file
EVENT UserEvent()
{
        //Continuous to process this event
}
```