



Rave Quick Reference

"SQL Function API Reference"


Document Version 1.0 November 2009	Medidata Solutions, Inc. Corporate Office 79 Fifth Avenue New York, N.Y. 10003 +1 212 918 1800
<p>Medidata Solutions, Inc. Proprietary - Medidata and Knowledge / Tech Transfer Clients.</p> <p>Information in this document is subject to change without notice. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including, but not limited to, photocopying and recording, for any purpose without the express permission of Medidata Solutions, Inc.</p> <p>Medidata Solutions Worldwide, Medidata CRO Contractor, Medidata Designer, Medidata Grants Manager, Medidata Rave, Medidata Services, Medidata University, and their respective logos are trademarks, or registered trademarks, of Medidata Solutions, Inc. All other brands or product names used in this document are trademarks, or registered trademarks, for their respective owners.</p> <p>Disclaimer: Given that Rave allows users to translate and change text strings, any screens shown in this document are only a representation of the product and may not match the actual interface you are currently using.</p> <div><p>Medidata Rave® 5.6.2 is certified by the Clinical Data Interchange Standards Consortium (CDISC) for Operational Data Modeling (ODM) standards ODM v1.2 and ODM v1.2.1.</p></div>	
© Copyright 2009 Medidata Solutions, Inc. All rights reserved.	

Table of Contents

1 Overview	4
1.1 Document Conventions	4
2 Rave SQL Functions	5
2.1 Table-Valued Functions	5
2.1.1 fnCompareObjectStatus	5
2.1.2 fnGlobalLibraryLoad.....	5
2.1.3 fnMergeLists.....	5
2.1.4 fnMergeThreeLists	6
2.1.5 fnMergeTwoLists	6
2.1.6 fnParseDelimitedString	7
2.1.7 fnParseParameterHeaders	7
2.1.8 fnProjectsLoad	8
2.1.9 fnSiteGroupChildrenRecursive	8
2.1.10 fnSplitStringInt	8
2.1.11 fnStudyUserSites	9
2.1.12 fnStudyUserSitesWithName	9
2.1.13 fnUsersByStudySite	10
2.2 Scalar-Valued Functions	11
2.2.1 fnAlpha.....	11
2.2.2 fnCheckMarkingExistsOnDataPoint	11
2.2.3 fnCheckUnitDictDefChange	11
2.2.4 fnCnvSQLDateTimeToRaveFmt	12
2.2.5 fnCodingGetPreferredTerm	12
2.2.6 fnCRFVersionDisplayIDName	13
2.2.7 fnDateFormat	13
2.2.8 fnDFNumberValue	14
2.2.9 fnDt	14
2.2.10 fnDtDD	14
2.2.11 fnDtMM.....	15
2.2.12 fnFlattenCodingValues	15
2.2.13 fnFlattenedInstanceName.....	15
2.2.14 fnFlattenFieldReviewGroups	16
2.2.15 fnFlattenReviewStatus	16
2.2.16 fnFlattenRoleSubjectStatus.....	16
2.2.17 fnFlattenTemplateFields	17
2.2.18 fnFloat.....	17
2.2.19 fnFormat.....	17
2.2.20 fnIsValidNumericCorrect.....	18
2.2.21 fnLDS.....	18
2.2.22 fnLocalDataString.....	19
2.2.23 fnLocalDefault	19

2.2.24 fnLocalizedDataPageName.....	19
2.2.25 fnLocalizedInstanceName	20
2.2.26 fnLocalizedRecordName	20
2.2.27 fnLocalString	21
2.2.28 fnPadLeft	21
2.2.29 fnUpperTrim.....	21

1 Overview

This document describes the SQL Functions that are part of the Rave database and can be used with Rave add-on applications and custom functions to retrieve, and work with, data.

1.1 Document Conventions

Code examples are formatted in Courier New:

```
private Client()  
{  
}
```

Items that will be replaced with a value are formatted in italics:

```
Client.SetLogLine(Message)
```

When there is a choice between one or more values, the items are enclosed in curly braces, and separated by the pipe "|" delimiter:

```
{True | False}
```

Menu names and commands, dialog box commands and controls; method, class, and interface names; and file and path names all use Verdana Bold:

When you click the **File > Open** command, a dialog box opens to allow you to select the file you want to open.

The root directory of the web site is located in
C:\inetpub\wwwroot

Optional items will be enclosed in square brackets. For example:

[optional item]

2 Rave SQL Functions

The SQL Functions described in this document can be used by Medidata developers or Knowledge/Tech. Transfer developers to extract data from Rave databases.

2.1 Table-Valued Functions

Table-Valued Functions return a database table containing the results.

2.1.1 fnCompareObjectStatus

Compares the status of two objects.

Syntax

```
fnCompareObjectStatus(@DataStatus1, @DataStatus2)
```

Returns

@Statuses TABLE with the following fields:

StatusName nvarchar(200)

Status1 int

Status2 int

Parameters

Parameter Name	Type	Description
@DataStatus1	int	ObjectID of the first object.
@DataStatus2	int	ObjectID for the second object to be compared.

2.1.2 fnGlobalLibraryLoad

Loads objects from the Global Library. This function returns a table containing the object data.

Syntax

```
@Objects = fnGlobalLibraryLoad()
```

Returns

@Objects TABLE containing the following fields:

ObjectID, int

ObjectNameID, int

ObjectNameString, nvarchar(2000)

Parameters

None

2.1.3 fnMergeLists

The function **fnMergeLists** returns a table, @Items, that contains the data from the three lists combined into a single table.

Syntax

```
@Items= fnMergeLists(@FirstList, @SecondList, @ThirdList)
```

Returns

@Items TABLE containing the following columns:

First nvarchar(50)

Second nvarchar(50)

Third nvarchar(50)

Parameters

Parameter Name	Type	Description
@FirstList	nvarchar(4000)	The data for the first list.
@SecondList	nvarchar(4000)	The data for the second list.
@ThirdList	nvarchar(4000)	The data for the third list.

2.1.4 fnMergeThreeLists

Merges data from the three lists and returns a single table.

Syntax

```
@Items = fnMergeThreeLists(@FirstList, @SecondList, @ThirdList)
```

Returns

@Items TABLE containing the following columns:

First nvarchar(50)

Second nvarchar(50)

Third nvarchar(50)

Parameters

Parameter Name	Type	Description
@FirstList	nvarchar(4000)	The data for the first list.
@SecondList	nvarchar(4000)	The data for the second list.
@ThirdList	nvarchar(4000)	The data for the third list.

2.1.5 fnMergeTwoLists

Combines the two lists passed to the function and returns a single table containing the combined data.

Syntax

```
@Items = fnMergeTwoLists(@FirstList, @SecondList)
```

Returns

@Items TABLE containing the following columns:

First nvarchar(50)

Second nvarchar(50)

Parameters

Parameter Name	Type	Description
@FirstList	nvarchar(4000)	The data for the first list.
@SecondList	nvarchar(4000)	The data for the second list.

2.1.6 fnParseDelimitedString

Breaks up a string at each instance of the delimiter character and inserts each piece into a row in the returned table. If the delimiter does not occur in the string, the whole string is returned in a single row.

Syntax

```
@Items = fnParseDelimitedString(@StringToParse, @Delimiter)
```

Returns

@Items TABLE with a single column:

Item nvarchar(4000)

Parameters

Parameter Name	Type	Description
@StringToParse	nvarchar(MAX)	The delimited string to be parsed.
@Delimiter	nchar(1)	The character used as the list delimiter.

2.1.7 fnParseParameterHeaders

Parses the parameters for each header from the Parameters table. The parsed values are inserted into the Parameters table.

Syntax

```
@Items = fnParseParameterHeaders()
```

Returns

@Items TABLE with the following columns:

HeaderID int

ParameterID int

Parameters

None

2.1.8 fnProjectsLoad

Loads the data for the objects in the current project.

Syntax

```
@Objects = fnProjectsLoad()
```

Returns

@Objects TABLE with the following columns:

ObjectID int

ObjectNameID int

ObjectNameString nvarchar(2000)

Parameters

None

2.1.9 fnSiteGroupChildrenRecursive

Returns all of the children in the **SiteGroup** hierarchy.

Syntax

```
@TBL = fnSiteGroupChildrenRecursive()
```

Returns

@TBL TABLE with the following columns:

ParentSiteGroupID int

ChildSiteGroupID int

Depth int

Parameters

None

2.1.10 fnSplitStringInt

Splits a comma-delimited string of integers and returns its integer parts in an in-memory table.

Syntax

```
@OutTable = fnSplitStringInt(@Input)
```

Returns

@OutTable TABLE with the following column:

ID int NOT NULL

Parameters

Argument	Type	Description
@Input	varchar(MAX)	The comma-delimited list of integer values.

2.1.11 fnStudyUserSites

Returns all Sites for the specified **Study**, **User**, **Role**, **SiteGroup**, and existing **SiteList**.

Syntax

```
@Tmp = fnStudyUserSites(@st, @uid, @rid, @sg)
```

Returns

@Tmp TABLE with the following columns:

SiteID int

StudySiteID int

Parameters

Argument	Type	Description
@st	int	The StudyID of the Study for which Site information is returned. Only sites that are part of the specified Study are returned.
@uid	int	The UserID for which Site information is needed. Only sites visible to the specified user are returned.
@rid	int	The RoleID for which Site information is needed. Only sites visible for the RoleID are returned.
@sg	varchar(20)	A comma-delimited list of values. The first value is the SiteGroupID of the Site Group for which information is needed. If the second value is a 0, child Site information will not be included, if a 1, then child Site information will also be included in the returned site list.

2.1.12 fnStudyUserSitesWithName

Returns a table listing **SiteID**, **StudySiteID**, and **SiteName** for all Sites for a particular **Study**, **User**, **Role**, **SiteGroup**, **SiteList**, and **Locale**.

Syntax

```
@Tbl = fnStudyUserSitesWithName(@st, @uid, @rid, @sg, @si, @locale)
```

Returns

@Tbl TABLE with the following columns:

SiteID int

StudySiteID int,

SiteName nvarchar(255)

Parameters

Argument	Type	Description
@st	int	The StudyID of the Study for which Site information is returned. Only sites that are part of the specified Study are returned.
@uid	int	The UserID for which Site information is needed. Only sites visible to the specified user are returned.
@rid	int	The RoleID for which Site information is needed. Only sites visible for the RoleID are returned.
@sg	varchar(20)	A comma-delimited list of values. The first value is the SiteGroupID of the Site Group for which information is needed. If the second value is a 0, child Site information will not be included, if a 1, then child Site information will also be included in the returned site list.
@si	varchar(4096)	The SiteID of the Site for which the Site Name are retrieved.
@locale	char(3)	The locale.

2.1.13 fnUsersByStudySite

This function returns a one-column table containing the **UserID** of Users that have access to the sites specified in the parameters. The return values in the table are restricted by @RoleID.

WARNING: This function is not currently used by Rave, and therefore is not validated. Use with extreme caution.

Syntax

```
@tmpfnUsersByStudySite = fnUsersByStudySite(@studyId, @siteGroup, @SiteGroup,
@siteList, @userId int, @roleId int)
```

Returns

@tmp_fnUsersByStudySite TABLE with a single column:

@UserID int

Argument	Type	Description
@StudyID	int	The StudyID for which the user information is needed.
@SiteGroup	varchar(20)	SiteGroup. For example, '1,1' means World and everything below.
@SiteList	varchar(max)	A comma-delimited list of SiteIDs.
@UserId	int	The UserID of the current user.
@RoleID	int	The RoleID of the current user. The return values will be restricted by the specified RoleID.

2.2 Scalar-Valued Functions

2.2.1 fnAlpha

Replaces each non-alphanumeric character in a string with an underline character (), and then returns the resulting string.

Syntax

```
@AlphaStr = fnAlpha(@instring)
```

Returns

@outstring nvarchar(255)

Argument	Type	Description	Default Value
@instring	nvarchar(255)	The string to be modified.	no default

Remarks

Valid values include: a-z, A-Z, 0-9, and \$

2.2.2 fnCheckMarkingExistsOnDataPoint

Determines whether there is a **Marking** on the specified **DataPoint**.

Syntax

```
@result = fnCheckMarkingExistsOnDataPoint(@DataPointID)
```

Returns

@result bit, 0 if the Marking does not exist for the specified DataPoint, 1 if there is a Marking on the specified DataPoint.

Argument	Type	Description
@DataPoint	int	The DataPointID of the DataPoint to check for Markings.

2.2.3 fnCheckUnitDictDefChange

Compares two unit dictionary entries, specified as sEntryID and tEntryID, and returns a value which indicates whether or not they are the same.

Syntax

```
@bit = fnCheckUnitDictDefChanges(@sEntryID, @tEntryID)
```

Returns

@bit bit, 0 if the entries are different, 1 if the two entries are the same.

Argument	Type	Description
@sEntryID	int	The first unit dictionary entry.

Argument	Type	Description
@tEntryID	int	The second unit dictionary entry.

2.2.4 fnCnvSQLDateTimeToRaveFmt

This function takes a SQL DateTime variable (a string representing a Rave date/time format and locale), and then converts the datetime, which is passed in the sql datetime variable, to a nvarchar(50) string in accordance with the specified Rave format.

Syntax

```
@newDateString = fnCnvSQLDateTimeToRaveFmt(@dt, @raveDateFormat, @loc)
```

Returns

@newDateString nvarchar(50)

Argument	Type	Description
@dt	datetime	The date to be formatted.
@raveDateFormat	varchar(50)	The Rave format to be used with the desired date value.
@loc	char(3)	The desired locale for the formatted date.

Remarks

Date Formats include any combination of:

- dd** = double digit date (e.g., 01, 06, 11, 21)
- mm** = double digit month (e.g., 03, 09, 10)
- MMM** = month abbreviation (e.g., MAR, SEP, OCT)
- yyyy** = four digit year
- yy** = two digit year
- hh** = 12-hour format double digit hour (e.g., 04)
- HH** = 24-hour format double digit hour (e.g., 16)
- nn** = minutes (e.g., 01, 09, 59)
- ss** = seconds (e.g., 03, 45)
- rr** = meridian (e.g., AM or PM)

as well as any of the following split characters: <space>, forward slash "/", colon ":", period "."

Optional tokens are demoted by a trailing hyphen "-", and must be followed by a split character. A trailing **hyphen** "-" after a date part results in an auto-fill of the configured default for that date part. For example, **UN** becomes acceptable for a **dd** value, which then translates to the default day of '01' or '15'. For derivation and calculation purposes, **UN** is the hard-coded unknown value for **dd-** or **mm-**, and **UNK** is the hard-coded unknown value for **MMM-**.

2.2.5 fnCodingGetPreferredTerm

Retrieves the preferred coded term for the specified FieldID.

Syntax

```
@CodedTerm = fncodingGetPreferredTerm(@FieldID, @RecordID)
```

Returns

@CodedTerm nvarchar(2000)

Argument	Type	Description
@FieldID	int	The FieldID of the field for which the preferred term is needed.
@RecordID	int	The RecordID of the record associated with the field.

Remarks

If the submitted field is a log field, then the function will perform the lookup on the current record. For non-log fields, recordposition 0 is used.

2.2.6 fnCRFVersionDisplayIDName

Returns a string that combines the CRF Version ID and the CRF Name.

Syntax

```
@return = fnCRFVersionDisplayIDName(@CrfVersionID, @Locale)
```

Returns

@return nvarchar(4000), a string that combines the CRFVersionID and CRFVersionName.

Argument	Type	Description
@CRFVersionID	int	The CRFVersionID for which the combined string is needed. "untiled.doc"

2.2.7 fnDateFormat

Returns a date string that has been formatted for the selected locale.

Syntax

```
@FormattedDate = fnDateFormat(@instring, @dateformat, @locale)
```

Returns

@FormattedDate, nvarchar(19), a string containing the formatted date value.

Argument	Type	Description
@InString	nvarchar(2000)	The date to be formatted.
@DateFormat	nvarchar(255)	The date format string.
@locale	nvarchar(3)	The locale in which the date is formatted.

2.2.8 fnDFNumberValue

Extracts an integer value from the supplied string.

Syntax

```
@retval = fnDFNumberValue(@str, @position)
```

Returns

@retval, int, an integer representation of the value passed into @Str.

Argument	Type	Description
@Str	varchar(200)	The string from which the number value is parsed.
@Position	char	Set position to 'D' or 'd' to indicate that the number to be extracted is a decimal.

Remarks

If the number is a decimal number, the value will be truncated at the decimal point.

2.2.9 fnDt

Returns a date value, formatted as yyyy-mm-dd hh:m:ss.

Syntax

```
@outDate = fnDt(@InString, @MonthIfMissing, @DayIfMissing)
```

Returns

@outDate datetime

Argument	Type	Description
@InString	char(5)	The name of the file that is associated with this object.
@MonthIfMissing	int	An integer representing the month value. It is only used if the month value is missing from the string.
@DayIfMissing	int	An integer representing the day of the month. It is only used if the day value is missing from the string.

2.2.10 fnDtDD

Extracts the day from the supplied date string and returns the value as an integer.

Syntax

```
@day = fnDtDD(@InString)
```

Returns

@day int value representing the day extracted from the date string.

Argument	Type	Description
@InString	nvarchar(2000)	A value representing a date. The day portion of the date will be extracted and returned as an integer.

2.2.11 fnDtMM

Extracts the month from the supplied date string and returns the value as an integer.

Syntax

```
@Month = fnDtMM(@InString)
```

Returns

@Month int, representing the month value extracted from the supplied date string.

Argument	Type	Description
@InString	nvarchar(2000)	A value representing a date. The month portion of the date will be extracted and returned as an integer.

2.2.12 fnFlattenCodingValues

Retrieves a string containing a comma-delimited list of Coding Entries with their associated **CodingEntryID** from the **CodingValues** table for the specified **RecordID** and **FileID**.

Syntax

```
@CodingEntries = fnFlattenCodingValues(@RecordID, @FileID)
```

Returns

@CodingEntires, nvarchar(200)

Argument	Type	Description
@RecordID	int	The RecordID for which coding values are extracted.
@FileID	int	The FileID for which coding values are extracted.

2.2.13 fnFlattenedInstanceName

Retrieves a string containing a flattened path to the specified FolderOID.

Syntax

```
@name = fnFlattenedInstanceName(@InstanceID)
```

Returns

@name nvarchar(400), a string representing the path to a particular folder.

Argument	Type	Description
@InstanceID	int	The InstanceID of the folder for which the flattened instance name is needed.

Remarks

The returned string contains the path in the following format: '*ParentFolderOID (n) – ParentFolderOID (n) – FolderOID (n)*' where ParentOID is any parent folder(s) between the specified folder and the top of the folder tree, and n represents the number of instances for the associated folder.

2.2.14 fnFlattenFieldReviewGroups

Retrieves a comma-delimited list of Review Groups for the supplied **FieldID**.

Syntax

```
@ReviewGroups = @fnFlattenFieldReviewGroups(@FieldID)
```

Returns

@ReviewGroups varchar(200), a string containing a comma-delimited list of Review Groups.

Argument	Type	Description
@FieldID	int	The FieldID of the field for which the flattened list of Review Groups is needed.

2.2.15 fnFlattenReviewStatus

Retrieves a comma-delimited list of the Review status for the selected **DataPointID**.

Syntax

```
@ReviewStatus = fnFlattenReviewStatus(@DataPointID)
```

Returns

@ReviewStatus nvarchar(50), a comma-delimited list of ReviewStatus and ReviewGroupID for the specified data point.

Argument	Type	Description
@DataPointID	int	The DataPointID of the data point for which the review status is retrieved.

2.2.16 fnFlattenRoleSubjectStatus

Retrieves a comma-delimited list of **SubjectStatusID** values for the specified **RoleID**.

Syntax

```
@ReturnString = fnFlattenRoleSubjectStatus(@RoleID)
```

Returns

@ReturnString varchar(200) a string containing the comma delimited list of **SubjectStatusID** values for the specified RoleID.

Argument	Type	Description
@RoleID	int	The RoleID for which the list of SubjectStatusID

Argument	Type	Description
values is needed.		

2.2.17 fnFlattenTemplateFields

Retrieves a comma-delimited list of FieldOID values along with the Visible status of those fields for the supplied TemplateID.

Syntax

```
@ReturnString = fnFlattenTemplateFields(@TemplateID)
```

Returns

@ReturnString varchar(500), a string containing a comma-delimited list of FieldOID values.

Argument	Type	Description
@TemplateID	int	The TemplateID of the template for which the list of FieldOID values is returned.

Remarks

The list consists of a FieldOID along with a single character indicating whether the field is visible or not. 1 indicates that the field is visible, and 0 indicates that the field is not visible.

2.2.18 fnFloat

Extracts a float value from a string.

Syntax

```
@NumberOut = fnFloat(@InString)
```

Returns

@NumberOut float

Argument	Type	Description
@InString	nvarchar(20)	The string containing the number to be extracted.

Remarks

This function will strip out the thousand separators, currency character, and plus sign from the string parameter.

2.2.19 fnFormat

Replaces {0}, {1}, {2}, etc. in the **@Format** parameter with the corresponding values from the **@Values** parameter.

Syntax

```
@FormattedString = fnFormat(@Format, @Values, @Delimiter)
```

Returns

@FormattedString nvarchar(4000), the string supplied in the @Format parameter with the values from the @Values parameter inserted into the placeholder characters.

Argument	Type	Description
@Format	nvarchar(2000)	The format string into which the values, in the @Values string, are inserted.
@Values	nvarchar(2000)	A list of values delimited by the character in the @Delimiter parameter.
@Delimiter	char	The character used as a delimiter in the list contained in the @Values parameter.

2.2.20 fnIsValidNumericCorrect

Determines whether the supplied string contains a valid numeric value.

Syntax

```
@IsNumber = fnIsValidNumericCorrect(@InString)
```

Returns

@IsNumber bit, 0 indicates that this string does not contain a valid number, and 1 indicates that the string does contain a valid number.

Argument	Type	Description
@InString	varchar(20)	The string to be tested to see if it contains a valid numeric value.

2.2.21 fnLDS

Returns the string associated with the supplied **StringID** for the desired location.

Syntax

```
@String = fnLDS(@StringID, @Locale)
```

Returns

@String nvarchar(4000), the localized string for the **StringID** and **Locale**.

Argument	Type	Description
@StringID	int	The StringID for which the localized string is returned.
@Locale	char(3)	The locale for which the string should be returned.

Remarks

If the string does not exist for the selected Locale, the string for the Default Locale, and for the **StringID**, will be returned instead.

2.2.22 fnLocalDataString

Returns the string associated with the supplied **StringID** for the desired location.

Syntax

```
@String = fnLDS(@StringID, @Locale)
```

Returns

@String nvarchar(4000), the localized string for the **StringID** and **Locale**.

Argument	Type	Description
@StringID	int	The StringID for which the localized string is returned.
@Locale	char(3)	The locale for which the string should be returned.

Remarks

Unlike fnLDS, this function does not check to see if the string exists for the requested locale and may return an empty string.

2.2.23 fnLocalDefault

Retrieves the localized string for the StringID of the default locale.

Syntax

```
@String = fnLocalDefault(@StringID)
```

Returns

Argument	Type	Description
@StringID	int	The StringID for which the localized string is returned.

2.2.24 fnLocalizedDataPageName

Returns the localized name for the supplied **DataPageID** and **Locale**.

Syntax

```
@DataPageName = fnLocalizedDataPageName(@Locale, @DataPageID)
```

Returns

@DataPageName nvarchar(4000)

Argument	Type	Description
@Locale	char(3)	The locale for which the data page name should be returned.
@DataPageID	int	The ID of the data page for which the localized name is needed.

Remarks

In addition to the localized **DataPageName** and **DataPageNumber**, the value returned includes the following localized values (if they exist for the selected locale): **FormName**, **FolderName**, **InstanceNumber**, **DataPageDate**, and **SubjectDate**.

2.2.25 fnLocalizedInstanceName

Retrieves the localized name for the supplied **InstanceID** and **Locale**.

Syntax

```
@InstanceName = fnLocalizedInstanceName(@Locale, @InstanceID)
```

Returns

@InstanceName nvarchar(4000) the localized name of the Instance.

Argument	Type	Description
@Local	char(3)	The locale for which the localized Instance name is needed.
@InstanceID	int	The InstanceID of the instance for which the localized name is needed.

Remarks

In addition to the localized **InstanceName** and **InstanceNumber**, the value returned includes the following localized values (if they exist for the selected locale): **FolderName**, **SubjectDate**, and **InstanceDate**.

2.2.26 fnLocalizedRecordName

Retrieves the localized name for the supplied **RecordID** and **Locale**.

Syntax

```
@RecordName = fnLocalizedRecordName(@Locale, @RecordID)
```

Returns

@RecordName nvarchar(4000)

Argument	Type	Description
@Local	char(3)	The locale for which the localized Instance name is needed.
@RecordID	int	The RecordID of the record for which the localized name is needed.

Remarks

In addition to the localized **RecordName**, the value returned will include the following localized values (if they exist for the selected locale): **RecordPosition**, **FormName**, **SubjectDate**, **InstanceNumber**, **FolderName**, **DatePageDate**, and **RecordDate**.

2.2.27 fnLocalizedString

Retrieves a localized string based on the string's name for the specified **Locale**.

Syntax

```
@ReturnString = fnLocalizedString(@Key, @Locale)
```

Returns

```
@ReturnString nvarchar(4000)
```

Argument	Type	Description
@Key	varchar(50)	The name of the string for which the localized version is needed.
@Locale	char(3)	The locale for which the localized string is needed.

Remarks

If the string does not exist for the selected Locale, this function returns the value passed in the @Key parameter.

2.2.28 fnPadLeft

Pads the specified string on the left to the specified length with the specified character.

Syntax

```
@ReturnString = @fnPadLeft(@BaseString, @PadChar, @Length)
```

Returns

```
@ReturnString nvarchar(2000)
```

Argument	Type	Description
@BaseString	nvarchar(2000)	The string which is padded on the left with the supplied character.
@PadChar	nchar	The character with which the string is padded.
@Length	int	The total length of the resulting string.

Remarks

If the length specified by the **@Length** parameter is greater than 2000 characters, or the current length of **@BaseString** is greater than the length requested, then the function returns **@BaseString**. If the length of **@BaseString** is less than @Length, the string will be padded on the left with enough copies of **@PadChar** to extend the length of **@BaseString** to **@Length**.

2.2.29 fnUpperTrim

Returns the supplied string converted into uppercase and trimmed of leading and trailing spaces.

Syntax

```
@ReturnString = fnUpperTrim(@Str)
```

Returns

@ReturnString varchar(500)

Argument	Type	Description
@Str	varchar(500)	The string which is converted to uppercase and trimmed of leading and trailing blanks.

Revision History

Version	Date	Description of Changes
1.0	13 Nov 2009	Original Release