Commence Dynamic Data Exchange Support

About the Commence DDE Help File

All information in this file was taken from the Commence manual and documents provided by Commence Corporation

Alphabetical List of Functions

AddItem	**[AddItem(*Category, Item, Clarify Value*)]**
AddSharedItem	**[AddSharedItem(*Category, Item*)]**
AppendText	**[AppendText(*Category, Item, Field, Text*)]**
AssignConnection	**[AssignConnection(*FromCategory, FromItem, ConnectionName, ToCategory, ToItem*)]**
CheckInFormScript	**[CheckInFormScript(*Category, FormName, Filename*)]**
CheckOutFormScript	**[CheckOutFormScript(*Category, FormName, Filename*)]**
ClarifyItemNames	**[ClarifyItemNames(*Status*)]**
Databases	**Databases**
DeleteItem	**[DeleteItem(*Category, Item*)]**
DeleteView	**[DeleteView(*View Name*)]**
EditItem	**[EditItem(*Category, Item, Field, FieldValue*)]**
FireTrigger	**[FireTrigger(*Trigger, Arg2..., Arg9*)]**
Formats	**Formats**
GetActiveViewInfo	**[GetActiveViewInfo()]**
GetCallerID	**[GetCallerID(*Category, PhoneNumber*)]**
GetCategoryCount	**[GetCategoryCount()]**
GetCategoryDefinition	**[GetCategoryDefinition(*Category*)]**
GetCategoryNames	**[GetCategoryNames()]**
GetConnectedItemCount	**[GetConnectedItemCount(*FromCategory, FromItem, Connection, ToCategory*)]**
GetConnectedItemField	**[GetConnectedItemField(*FromCategory, FromItem*, Connection, ToCategory, Field [, Delimiter])]**
GetConnectedItemNames	**[GetConnectedItemNames(*FromCategory, FromItem, Connection, ToCategory*)]**
GetConnectionCount	**[GetConnectionCount(*Category*)]**
GetConnectionNames	**[GetConnectionNames(*Category*)]**
GetDatabase	**[GetDatabase()]**
GetDatabaseDefinition	**[GetDatabaseDefinition()]**
GetDesktopCount	**[GetDesktopCount()]**
GetDesktopNames	**[GetDesktopNames()]**
GetField	**[GetField(*Category, Item, Field*)]**
GetFieldCount	**[GetFieldCount(*Category*)]**
GetFieldDefinition	**[GetFieldDefinition(*Category, Field*)]**
GetFieldNames	**[GetFieldNames(*Category*)]**
GetFields	**[GetFields(*Category, Item, n, Field_1..., Field_n*)]**
GetFieldToFile	**[GetFieldToFile(*Category, Item, Field, Filename*)]**
GetFormCount	**[GetFormCount(*Category*)]**
GetFormNames	**[GetFormNames(*Category*)]**
GetImageFieldNames	**[GetImageFieldNames(Category, Delim)]**
GetImageFieldCount	**[GetImageFieldCount(Category)]**
GetImageFieldToFile	**[GetImageFieldToFile(*Category, Item, Field, Filename*)]**
GetItemCount	**[GetItemCount(*Category*)]**
GetItemNames	**[GetItemNames(*Category*)]**
GetLastError	**[GetLastError]**
GetLetterViewInfo	**[GetLetterViewInfo()]**
GetMarkItem	**[GetMarkItem(*Category, Item, Clarify Value*)]**
GetMeField	**[GetImageFieldCount(*Category*)]**
GetPhoneNumber	**[GetPhoneNumber(*PhoneNumber*)]**
GetPreference	**[GetPreference(*Setting*)]**
GetReverseName	**[GetReverseName(*Name, PrefFlag*)]**
GetTriggerCount	**[GetTriggerCount()]**
GetTriggerNames	**[GetTriggerNames()]**
GetViewCount	**[GetViewCount(*Category*)]**
GetViewNames	**[GetViewNames(*Category*)]**

GetViewToFile	**[GetViewToFile (*Viewname, Mode, Param1, Param2, Filename*)]**
LogPhoneCall	**[LogPhoneCall(*Category1, Item_1..., Category_n, Item_n*)]**
MarkActiveItem	**[MarkActiveItem]**
MergeTemplateCreate	**[MergeTemplateCreate(*name, Category, Shared*)]**
MergeTemplateSave	**[MergeTemplateSave(*name, Shared*)]**
PromoteItemToShared	**[PromoteItemToShared(*Category, Item*)]**
ShowDesktop	**[ShowDesktop(*Desktop Name*)]**
ShowItem	**[ShowItem(*Category, Item*)]**
ShowView	**[ShowView(*View Name, 1*)]**
Status	**Status**
SysItems	**SysItems**
Topics	**Topics**
UnassignConnection	**[UnassignConnection(*FromCategory, FromItem, ConnectionName, ToCategory, ToItem*)]**
Version	**Version**
VersionExtended	**VersionExtended**
ViewCategory	**[ViewCategory(*Category*)]**
ViewConjunction	**[ViewConjunction(*AndOr12, AndOr13, AndOr34*)]**
ViewConnectedCount	**[ViewConnectedCount(*Index, Connection, ToCategory*)]**
ViewConnectedField	**[ViewConnectedField(*Index, ConnectionName, ToCategory, ConnIndex, Field*)]**
ViewConnectedFields	**[ViewConnectedFields(*Index, ConnectionName, ToCategory, ConnIndex, n, Field_1..., Field_n*)]**
ViewConnectedItem	**[ViewConnectedItem(*Index, Connection, ToCategory, ConnIndex*)]**
ViewDeleteAllItems	**[ViewDeleteAllItems()]**
ViewField	**[ViewField(*Index, Field*)]**
ViewFields	**[ViewFields(*Index, n, Field_1..., Field_n*)]**
ViewFieldToFile	**[ViewFieldToFile(*Index, Field, Filename*)]**
ViewFilter	**[ViewFilter(*ClauseNumber, FilterType, NotFlag, FieldTypeParameters...*)]**
ViewImageFieldToFile	**[ViewImageFieldToFile(*Index, Field, Filename*)]**
ViewItemCount	**[ViewItemCount()]**
ViewItemIndex	**[ViewItemIndex(*NameFieldValue*)]**
ViewItemName	**[ViewItemName(*Index*)]**
ViewMarkItem	**[ViewMarkItem(*Index*)]**
ViewReverseName	**[ViewReverseName(*Name, PrefFlag*)]**
ViewSaveView	**[ViewSaveView(*New View Name, Shared*)]**
ViewSort	**[ViewSort(*Field1, Sort1, Field2, Sort2 , Field3, Sort3 , Field4, Sort4*)]**
ViewView	**[ViewView(*View Name*)]**

Note: Blue text denotes optional parameters.

Note: Dark red text denotes parameters which may be left blank. Default values will be used.

DDE commands by function groups

Commands that return information about a Category

GetCategoryCount	[GetCategoryCount()]
GetCategoryDefinition	[GetCategoryDefinition(*Category*)]
GetCategoryNames	[GetCategoryNames()]
GetConnectionCount	[GetConnectionCount(*Category*)]
GetConnectionNames	[GetConnectionNames(*Category*)]
GetFieldCount	[GetFieldCount(*Category*)]
GetFieldNames	[GetFieldNames(*Category*)]
GetFormCount	[GetFormCount(*Category*)]
GetFormNames	[GetFormNames(*Category*)]
GetImageFieldNames	[GetImageFieldNames(*Category, Delim*)]
GetImageFieldCount	[GetImageFieldCount(*Category*)]

Commands that act on a View

DeleteView	[DeleteView(*View Name*)]
GetActiveViewInfo	[GetActiveViewInfo()]
GetLetterViewInfo	[GetLetterViewInfo()]
GetViewCount	[GetViewCount(*Category*)]
GetViewToFile	[GetViewToFile (*Viewname, Mode, Param1, Param2, Filename*)]
GetViewNames	[GetViewNames(*Category*)]
ViewConjunction	[ViewConjunction(*AndOr12, AndOr13, AndOr34*)]
ViewFilter	[ViewFilter(*ClauseNumber, FilterType, NotFlag, FieldTypeParameters...*)]
ViewSaveView	[ViewSaveView(*New View Name, Shared*)]
ViewSort	[ViewSort(*Field1, Sort1, Field2, Sort2 , Field3, Sort3 , Field4, Sort4*)]
ViewView	[ViewView(*View Name*)]
ShowView	[ShowView(*View Name, 1*)]

Commands that return information about an Item

GetConnectedItemCount	[GetConnectedItemCount(*FromCategory, FromItem, Connection, ToCategory*)]
~~**GetConnectedItemField**~~**GetConnectedItemField**	[GetConnectedItemField(FromCategory, *FromItem,* Connection, ToCategory, Field [, Delimiter])]
~~**GetConnectedItemNames**~~**GetConnectedItemNames**	[GetConnectedItemNames(FromCategory, *FromItem, Connection, ToCategory)]*
GetItemCount	[GetItemCount(*Category*)]
GetMeField	[GetImageFieldCount(*Category*)]
ViewConnectedCount	[ViewConnectedCount(*Index, Connection, ToCategory*)]
ViewConnectedItem	[ViewConnectedItem(*Index, Connection, ToCategory, ConnIndex*)]
ViewItemCount	[ViewItemCount()]

Commands that return Field Values

GetConnectedItemField	[GetConnectedItemField(FromCategory, *FromItem,* Connection, ToCategory, Field [, Delimiter])]
GetConnectedItemNames	[GetConnectedItemNames(FromCategory, *FromItem, Connection, ToCategory)]*
GetField	[GetField(*Category, Item, Field*)]
GetFields	[GetFields(*Category, Item, n, Field_1..., Field_n*)]
GetFieldToFile	[GetFieldToFile(*Category, Item, Field, Filename*)]
GetImageFieldNames	**[GetImageFieldNames(Category, Delim)]**
GetImageFieldCount	**[GetImageFieldCount(Category)]**
GetImageFieldToFile	**[GetImageFieldToFile(*Category, Item, Field, Filename*)]**
GetItemNames	[GetItemNames(*Category*)]
GetPhoneNumber	[GetPhoneNumber(*PhoneNumber*)]
GetReverseName	[GetReverseName(*Name, PrefFlag*)]
ViewConnectedField	[ViewConnectedField(*Index, ConnectionName, ToCategory, ConnIndex, Field*)]
ViewConnectedFields	[ViewConnectedFields(*Index, ConnectionName, ToCategory, ConnIndex, n, Field_1..., Field_n*)]
ViewField	[ViewField(*Index, Field*)]
ViewFields	[ViewFields(*Index, n, Field_1..., Field_n*)]
ViewFieldToFile	[ViewFieldToFile(*Index, Field, Filename*)]
ViewImageFieldToFile	[ViewImageFieldToFile(*Index, Field, Filename*)]
ViewItemIndex	[ViewItemIndex(*NameFieldValue*)]
ViewItemName	[ViewItemName(*Index*)]
ViewReverseName	[ViewReverseName(*Name, PrefFlag*)]

DDE commands by topic

- Request Items for the System Topic
- Request Items
- Execute Items

Request Items for the System Topic

Databases	Databases
Formats	Formats
Status	Status
SysItems	SysItems
Topics	Topics
Version	Version
VersionExtended	VersionExtended

Request Items

ClarifyItemNames	[ClarifyItemNames(*Status*)]
GetActiveViewInfo	[GetActiveViewInfo()]
GetCallerID	[GetCallerID(*Category, PhoneNumber*)]
GetCategoryCount	[GetCategoryCount()]
GetCategoryNames	[GetCategoryNames()]
GetCategoryDefinition	[GetCategoryDefinition(*Category*)]
GetConnectedItemCount	[GetConnectedItemCount(*FromCategory, FromItem, Connection, ToCategory*)]
GetConnectionCount	[GetConnectionCount(*Category*)]
GetConnectionNames	[GetConnectionNames(*Category*)]
GetConnectedItemField	**[GetConnectedItemField(*FromCategory, FromItem*, Connection, ToCategory, Field [, Delimiter])]**
GetConnectedItemNames	[GetConnectedItemNames(*FromCategory, FromItem, Connection, ToCategory*)]
GetDatabase	[GetDatabase()]
GetDesktopCount	[GetDesktopCount()]
GetDesktopNames	[GetDesktopNames()]
GetField	[GetField(*Category, Item, Field*)]
GetFieldCount	[GetFieldCount(*Category*)]
GetFieldNames	[GetFieldNames(*Category*)]
GetFieldDefinition	[GetFieldDefinition(*Category, Field*)]
GetFields	[GetFields(*Category, Item, n, Field_1..., Field_n*)]
GetFieldToFile	[GetFieldToFile(*Category, Item, Field, Filename*)]
GetFormCount	[GetFormCount(*Category*)]
GetFormNames	[GetFormNames(*Category*)]
GetImageFieldNames	**[GetImageFieldNames(Category, Delim)]**
GetImageFieldCount	**[GetImageFieldCount(Category)]**
GetImageFieldToFile	[GetImageFieldToFile(*Category, Item, Field, Filename*)]
GetItemCount	[GetItemCount(*Category*)]
GetItemNames	[GetItemNames(*Category*)]
GetLastError	[GetLastError]
GetLetterViewInfo	[GetLetterViewInfo()]
GetMarkItem	[GetMarkItem(*Category, Item, Clarify Value*)]
GetMeField	[GetImageFieldCount(*Category*)]
GetPhoneNumber	[GetPhoneNumber(*PhoneNumber*)]
GetReverseName	[GetReverseName(*Name, PrefFlag*)]
GetTriggerCount	[GetTriggerCount()]
GetTriggerNames	[GetTriggerNames()]
GetViewCount	[GetViewCount(*Category*)]
GetViewNames	[GetViewNames(*Category*)]
GetViewToFile	[GetViewToFile (*Viewname, Mode, Param1, Param2, Filename*)]
MarkActiveItem	[MarkActiveItem]
MergeTemplateCreate	[MergeTemplateCreate(*name, Category, Shared*)]
MergeTemplateSave	[MergeTemplateSave(*name, Shared*)]
ViewCategory	[ViewCategory(*Category*)]
ViewConjunction	[ViewConjunction(*AndOr12, AndOr13, AndOr34*)]
ViewConnectedCount	[ViewConnectedCount(*Index, Connection, ToCategory*)]
ViewConnectedField	[ViewConnectedField(*Index, ConnectionName, ToCategory, ConnIndex, Field*)]
ViewConnectedFields	[ViewConnectedFields(*Index, ConnectionName, ToCategory, ConnIndex, n, Field_1..., Field_n*)]
ViewConnectedItem	[ViewConnectedItem(*Index, Connection, ToCategory, ConnIndex*)]
ViewDeleteAllItems	[ViewDeleteAllItems()]
ViewField	[ViewField(*Index, Field*)]
ViewFields	[ViewFields(*Index, n, Field_1..., Field_n*)]
ViewFieldToFile	[ViewFieldToFile(*Index, Field, Filename*)]

ViewFilter	[ViewFilter(*ClauseNumber, FilterType, NotFlag, FieldTypeParameters...*)]
ViewImageFieldToFile	[ViewImageFieldToFile(*Index, Field, Filename*)]
ViewItemCount	[ViewItemCount()]
ViewItemName	[ViewItemName(*Index*)]
ViewItemIndex	[ViewItemIndex(*NameFieldValue*)]
ViewMarkItem	[ViewMarkItem(*Index*)]
ViewReverseName	[ViewReverseName(*Name, PrefFlag*)]
ViewSaveView	[ViewSaveView(*New View Name, Shared*)]
ViewSort	[ViewSort(*Field1, Sort1, Field2, Sort2 , Field3, Sort3 , Field4, Sort4*)]
ViewView	[ViewView(*View Name*)]

Execute Items

AddItem	[AddItem(*Category, Item, Clarify Value*)]
AddSharedItem	[AddSharedItem(*Category, Item*)]
AppendText	[AppendText(*Category, Item, Field, Text*)]
AssignConnection	[AssignConnection(*FromCategory, FromItem, ConnectionName, ToCategory, ToItem*)]
CheckInFormScript	[CheckInFormScript(*Category, FormName, Filename*)]
CheckOutFormScript	[CheckOutFormScript(*Category, FormName, Filename*)]
DeleteItem	[DeleteItem(*Category, Item*)]
DeleteView	[DeleteView(*View Name*)]
EditItem	[EditItem(*Category, Item, Field, FieldValue*)]
FireTrigger	[FireTrigger(*Trigger, Arg2..., Arg9*)]
GetViewToFile	**[GetViewToFile (Viewname, Mode, Param1, Param2, Filename)]**
LogPhoneCall	[LogPhoneCall(*Category1, Item_1..., Category_n, Item_n*)]
MergeTemplateCreate	**[MergeTemplateCreate(name, Category, Shared)]**
MergeTemplateSave	**[MergeTemplateSave(name, Shared)]**
PromoteItemToShared	[PromoteItemToShared(*Category, Item*)]
ShowDesktop	[ShowDesktop(*Desktop Name*)]
ShowItem	[ShowItem(*Category, Item*)]
ShowView	[ShowView(*View Name, 1*)]
UnassignConnection	[UnassignConnection(*FromCategory, FromItem, ConnectionName, ToCategory, ToItem*)]

See Also:

Miscellaneous Execute Notes

Error Codes for Request and Execute

If the NACK error code is less than 100, then that number indicates the position of the parameter that was invalid. For example, if the EXECUTE command [EditItem("Person", "John Doe", "Work Phone", "(123)555-1234")] returned a NACK error code of 2, then perhaps a John Doe does not exist in the Person category.

100	(0x64)	Out of memory
101	(0x65)	Internal error
102	(0x66)	Wrong number of parameters
103	(0x67)	Unknown conversation topic
104	(0x68)	No Phone Log category has been set; returned by LogPhoneCall EXECUTE command. Use **Customize-Preferences-Event Logs** to select a category.
105	(0x69)	No category selected; returned by ViewData REQUESTS not preceded by a ViewCategory.
106	(0x6A)	Unsupported clipboard format
107	(0x6B)	Unsupported field type
108	(0x6C)	Field/qualifier mismatch; returned by ViewFilter REQUEST
109	(0x6D)	Unknown EXECUTE command
110	(0x6E)	Parsing error. Check syntax of DDE command.
111	(0x6F)	Unknown REQUEST item or parsing error
112	(0x70)	Cannot add new item, category full; returned by AddItem, AddSharedItem and LogPhoneCall EXECUTE commands.
113	(0x71)	File I/O error. Is the disk full?
114	(0x72)	Connection already exists; returned by the AssignConnection EXECUTE command
115	(0x73)	No such connection exists; returned by the UnassignConnection EXECUTE commands
116	(0x74)	The category has been invalidated
117	(0x75)	The item has been invalidated.
118	(0x76)	An "every" date was received, or is the default value for a date field. Date ranges are not supported via DDE; returned by the AddItem or EditItem EXECUTE commands
119	(0x77)	Non-unique item name; returned by the AddItem or EditItem EXECUTE commands
120	(0x78)	Parameter too long; returned by the AddItem, EditItem and AppendText EXECUTE commands
121	(0x79)	No active child window or child window is unsupported; returned by the GetActiveViewInfo and GetLetterViewInfo REQUESTS
122	(0x80)	Agents are currently disabled. Returned by FireTrigger.
123	(0x81)	No item has been marked. Use ViewMarkItem or AddItem to mark an item.
124	(0x82)	Incompatible image type. Returned by GetImageFieldToFile or ViewImageFieldToFile.
125	(0x83)	Permission denied. Workgroup client does not have correct permission level.
126	(0x84)	No (-Me-) item has been defined. Use **Customize-Preferences-Personal Info**.
127	(0x85)	TAPI error encountered.
128	(0x86)	An agent exists but it is currently inactive. Returned by FireTrigger.
201	(0xC9)	Filter 1 has been invalidated; this may occur if a **ToCategory** item is deleted/modified
202	(0xCA)	Filter 2 has been invalidated
203	(0xCB)	Filter 3 has been invalidated
204	(0xCC)	Filter 4 has been invalidated

Definitions

AddItem

Syntax: [AddItem(*Category, Item, Clarify Value*)]

Execute Item for GetData and ViewData topics

Adds the indicated *Item* to the *Category*. Fields other than the Name field are set to their default values. (Default values for date fields must not specify date ranges, e.g., "every day"). By default, the new item is **local** (i.e., non-shared)

CAUTION: If the *Category* has mandatory fields, those fields must be filled in with subsequent EditItem commands. Use the GetFieldDefinition request to determine which fields are mandatory. Unpredictable results may follow if mandatory fields are left unfilled.

Note: If a clarify field is defined for the category (but is not a *Sequence Number* field), the clarify field value can be specified using the optional third parameter, *Clarify Value*. Alternatively, the *Item* parameter can be a clarified item name.

If the clarify field is a *Sequence Number* field, then the field value is determined automatically by Commence. An error results if the AddItem REQUEST tries to set a *Sequence Number* field. Use GetFieldDefinition to determine the *Sequence Number* field value so a clarified item name can be used with subsequent EditItems.

See Also:

Execute

Parameter Notes

AddSharedItem

Syntax: [AddSharedItem(*Category, Item*)]

Execute Item for GetData and ViewData topics

Identical in function to AddItem except that it creates a **shared** item, provided that the database is connected and the category is shared. Otherwise a local item is created.

CAUTION: If the *Category* has mandatory fields, those fields must be filled in with subsequent EditItem commands. Use the GetFieldDefinition request to determine which fields are mandatory. Unpredictable results may follow if mandatory fields are left unfilled.

See Also:

Execute

Parameter Notes

AppendText

Syntax: [AppendText(*Category, Item, Field, Text*)]

Execute Item for GetData and ViewData topics

Appends *Text* to an existing text *Field*. Use this command to overcome the 256 character maximum string limitation of certain macro languages (such as WordBasic). Whenever possible, use a clarified item name.

See Also:

Execute

Parameter Notes

AssignConnection

Syntax: [AssignConnection(*FromCategory, FromItem, Connection, ToCategory, ToItem*)]

Execute Item for GetData and ViewData topics

Assign a connection between the two indicated *Items* via *Connection*. Whenever possible, use a clarified item name.

See Also:

Execute

Parameter Notes

CheckInFormScript

Syntax: [CheckInFormScript(*Category, FormName, Filename*)]

Execute Item for GetData and ViewData topics

Updates the named detail form with the VBScript in the specified *Filename*. Only text (.txt) format files are currently supported.

Filename must be a fully qualified path including the drive letter (e.g. c:\tmp\script.txt).

CheckOutFormScript

Syntax: [CheckOutFormScript(*Category, FormName, Filename*)]

Execute Item for GetData and ViewData topics

Saves the VBScript associated with a particular detail form, *FormName*, to the named file, *Filename*. Only text (.txt) format files are currently supported.

Filename must be a fully qualified path including the drive letter (e.g. c:\tmp\script.txt).

ClarifyItemNames

Syntax: [ClarifyItemNames(*Status*)]

Request Item for the GetData topic

Tells Commence if clarified item names should be returned.

If *Status* is "True", Commence will always return clarified item names (if the category has a clarify field defined).

If *Status* is "False", non-clarified item names are returned. The default is "False".

If *Status* is left blank then Commence returns “True” or “False” to report back the present status.

Databases

Syntax: Databases

Request Item for the System topic

Returns a list of Commence databases. The list is of the form:

{Database Name 1}TAB**{Database Path 1}**CR/LF**{Database Name 2}**TAB**{Database Path 2}**CR/LF**...** CR/LF**{Database Name n}**TAB**{Database Path n}**

Note: Use the GetDatabase request to determine the currently active database.

DeleteItem

Syntax: [DeleteItem(*Category, Item*)]

Execute Item for GetData and ViewData topics

Deletes the indicated *Item* from the *Category*. Once an item is deleted, it cannot be recovered. Whenever possible, use a clarified item name..

Use with care!!

See Also:

Execute

Parameter Notes

DeleteView

Syntax: **[DeleteView(*View Name*)]**

Execute Item for GetData and ViewData topics

This command deletes the saved view.

Use with care!!

EditItem

Syntax: [EditItem(*Category, Item, Field, Value*)]

Execute Item for GetData and ViewData topics

Sets the value of *Field* to *Value* for the item identified by *Item* in *Category*. *Field* must not specify a Calculation field. Date fields may not specify a range, (e.g., "every day"). Whenever possible, use a clarified *Item* name.

If *Field* is an image field, *Value* must specify the filename containing the image. The filename must be a fully qualified path including the drive letter. The specified .bmp file is pasted into the image field. Only .BMP format image fields are currently supported. For example:

```
[EditItem(Person,”Doe, John”,Snapshot,c:\image\photo.bmp)]
```

See Also:

Execute

Parameter Notes

FireTrigger

Syntax: [FireTrigger(*Trigger*, *Arg2*,..*Arg9*)]

Execute Item for GetData and ViewData topics

This command fires the indicated Agent *Trigger*.

See Also:

Fire Trigger Notes

Formats

Syntax: Formats

Request Item for the System topic

Returns a tab-delimited list of clipboard formats supported by Commence for DDE.

GetActiveViewInfo

Syntax: [GetActiveViewInfo(*Delim*)]

Request Item for the GetData topic

Returns information about the currently active view. The return string is of the form:

{ViewName}Delim **{ViewType}**Delim **{CategoryName}**Delim **{ItemName}**Delim **{FieldName}**

{ViewType} can be any one of the following:

Book

Calendar

Gantt Chart

Report

Item Detail

{FieldName} is only valid if the active child window is an item detail. Note that a GetField on an active item detail will only retrieve the previous field value (before the edit). Until an item is saved, there is no way to retrieve updated field values.

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

GetCallerID

Syntax: [GetCallerID(*Category*, *Phone Number*,*Delim*)]

Request Item for the GetData topic

Returns a list of all items in the *Category* having the given phone number in a phone field. If *Category* is blank, searches all categories in the currently active database.

The phone number may be given in a variety of formats. Some examples of valid phone values are:

1(800)CALLNOW

415-555-1212

123

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

GetCategoryCount

Syntax: [GetCategoryCount]

Request Item for the GetData topic

Returns the number of categories defined in Commence.

GetCategoryDefinition

Syntax: [GetCategoryDefinition(*Category*,*Delim*)]

Request Item for the GetData topic

Returns a list with configuration information for the specified category.

The list is of the form:

{MaxItems}Delim **000000{S}{M}{D}{C}**Delim**{ClarifySeparator}**Delim**{ClarifyField}**

{MaxItems} is the maximum number of items for the category and is always 32000.

{S} is 1 for shared categories, 0 for local.

{M} is always 0.

{D} is 1 if duplicate item names are allowed, 0 otherwise.

{C} is 1 if a clarify field is defined, 0 otherwise.

If **{C}** is 1, the clarify separator is returned, followed by the clarify field name.

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

GetCategoryNames

Syntax: [GetCategoryNames(*Delim*)]

Request Item for the GetData topic

Returns a list of all categories found in Commence.

The list is of the form:

{Category 1}Delim **{Category 2}**Delim....Delim**{Category n}**

where **{Category n}** is the name of the nth category.

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

GetConnectedItemCount

Syntax: [GetConnectedItemCount(*FromCategory, FromItem, Connection, ToCategory*)]

Request Item for the GetData topic

Returns the number of items in *ToCategory* that *FromItem* of the *FromCategory* is connected to via *Connection*.

See Also:

Parameter Notes

GetConnectedItemField

Syntax: [GetConnectedItemField(*FromCategory, FromItem*, *Connection*, *ToCategory*, Field [, Delimiter])]

Request Item for the GetData topic

Returns the indirect field value(s) in ToCategory that FromItem of the FromCategory is connected to via Connection.

The order of the items in the list is arbitrary (i.e., not sorted).

The list is of the form:

{Connected Field Value 1}Delim { Connected Field Value 2}Delim....Delim{ Connected Field Value Item n}

where {Connected Field Value n} is the field value from the nth connected item.

Note: If the optional Delim parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The Delim parameter can be any string (up to 8 characters).

Text limitations:

·Limit to 256 chars

·First line of multiline field

·Returns string “(none)” if there are no connected items

See Also:

Parameter Notes

GetConnectedItemNames

Syntax: [GetConnectedItemNames(*FromCategory, FromItem, Connection, ToCategory, Delim*)]

Request Item for the GetData topic

Returns the list of items in *ToCategory* that *FromItem* of the *FromCategory* is connected to via *Connection.* The order of the items in the list is arbitrary (i.e., not sorted).

The list is of the form:

{Item 1}Delim **{Item 2}**Delim**....**Delim**{Item n}**

where **{Item n}** is the name of the nth item.

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

See Also:

Parameter Notes

GetConnectionCount

Syntax: [GetConnectionCount(*Category*)]

Request Item for the GetData topic

Returns the number of connections where the named Commence category is the **From** category.

GetConnectionNames

Syntax: [GetConnectionNames(*Category*, *Delim*, *ConnCatDelim*)]

Request Item for the GetData topic

Returns a list of all connections where the named Commence category is the **From** category.

The list is of the form:

{Connection 1}ConnCatDelim **{ToCategory 1}**Delim **{Connection 2}**ConnCatDelim **{ToCategory 2}**Delim**...**Delim**{Connection n}**ConnCatDelim **{ToCategory n}**

where **{Connection n}** is the name of the nth connection and **{ToCategory n}** is the name of the **To** category for the nth connection.

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

If the optional **ConnCatDelim** parameter is not specified, the connection names and their **To** categories are delimited with CR/LF (ASCII codes 13 and 10). The **ConnCatDelim** parameter can be any string (up to 8 characters).

GetDatabase

Syntax: [GetDatabase(*Delim*)]

Request Item for the GetData topic

Returns the name and path of the currently active database.

The return string is of the form:

{DatabaseName}Delim **{DatabasePath}**

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

GetDatabaseDefinition

Syntax: [GetDatabaseDefinition(*Delim*)]

Request Item for the GetData topic

Returns information about the curently active database. The returned string is of the form:

{DatabaseName}Delim **{DatabasePath}**Delim **000000{A}{X}{S}{C}**Delim **{UserName}**Delim **{SpoolPath}**

{DatabaseName} is the database name from the **Manage Database** dialog box.

{DatabasePath} is the full path of the database.

{A} is 1 if the database is attached, 0 otherwise.

{X} is 1 if the database is connected, 0 otherwise.

{S} is 1 if the database is a server, 0 otherwise.

{C} is 1 if the database is a client, 0 otherwise.

{UserName} is the user's login name.

{SpoolPath} is the full path to the server 'spool' area.

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

GetDesktopCount

Syntax: [GetDesktopCount]

Request Item for the GetData topic

Returns the number of desktops defined in the database.

GetDesktopNames

Syntax: [GetDesktopNames(*Delim*)]

Request Item for the GetData topic

Returns a list of all desktops found in Commence.

The list is of the form:

{Desktop 1}Delim **{Desktop 2}**Delim....Delim**{Desktop n}**

where **{Desktop n}** is the name of the nth desktop.

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

GetField

Syntax: [GetField(*Category, Item, Field*)]

Request Item for the GetData topic

Returns the data specified by the named Commence category, item and field.

FieldName cannot be of the field type Image.

See Also:

Parameter Notes

GetFieldCount

Syntax: [GetFieldCount(*Category*)]

Request Item for the GetData topic

Returns the number of fields defined in the named Commence category.

GetFieldDefinition

Syntax: [GetFieldDefinition(*Category, Field, Delim*)]

Request Item for the GetData topic

Returns a list with configuration information for the specified field.

The list is of the form:

{FieldType}Delim **000000{C}{S}{M}{R}**Delim **{MaxChars}**Delim **{DefaultString}**

{FieldType} indicates the field type and is one of the following values:

0 Text
1 Number
2 Date
3 Telephone
7 Check Box
11 Name
12 Data File
13 Image
14 Time
15 Excel Cell
20 Calculation
21 Sequence
22 Selection
23 E-Mail Address
24 Internet Address

{S} is 1 if the field is a shared, 0 if it is local.

{M} is 1 if the field is mandatory, 0 otherwise.

{R} is 1 if the field is a recurring date field, 0 otherwise.

{C} is 1 for a Selection field defined as a combobox, 0 otherwise.

{MaxChars} is the maximum size of the field.

{DefaultString} is the default value for the field.

Note: For a Sequence field, the **{DefaultString}** is the next sequence value that will be used.

If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

GetFieldNames

Syntax: [GetFieldNames(*Category, Delim*)]

Request Item for the GetData topic

Returns a list of fields found in the named Commence category.

The list is of the form:

{Field 1}Delim **{Field 2}**Delim....Delim**{Field n}**

where **{Field n}** is the name of the nth field.

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

GetFields

Syntax: [GetFields(*Category, Item, n, Field_1, Field_2 ,..., Field_n, Delim*)]

Request Item for the GetData topic

Returns multiple field values (*n*) for the named category and item. Image type fields are not allowed in the *Field_* parameters.

The returned data is of the form:

{Field 1}Delim **{Field 2}**Delim....Delim**{Field n}**

where **{Field n}** is the name of the nth field.

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

See Also:

Parameter Notes

GetFieldToFile

Syntax: [GetFieldToFile(*Category, Item, Field, Filename*)]

Request Item for the GetData topic

Identical to GetField except the data is copied to the specified *Filename*. If the file does not exist, Commence will create it; the drive and directory, however, must already exist. If the file exists, it will be overwritten. It is the client's responsibility to delete the file. The data is written out in text format, with paragraphs delimited with CR/LF (ASCII codes 13 and 10).

See Also:

Parameter Notes

GetFormCount

Syntax: [GetFormCount(*Category*)]

Request Item for the GetData topic

Returns the number of detail forms defined in the named Commence category.

GetFormNames

Syntax: [GetFormNames(*Category, Delim*)]

Request Item for the GetData topic

Returns a list of detail forms found in the named Commence category.

The list is of the form:

{Form 1}Delim **{Form 2}**Delim....Delim**{Form n}**

where **{Form n}** is the name of the nth detail form.

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

GetImageFieldCount

Syntax: [GetImageFieldCount(*Category*)]

Request Item for the GetData topic

Returns the number of images fields defined in the named Commence category.

GetImageFieldNames

Syntax: [GetImageFieldNames(*Category*, *Delim*)]

Request Item for the GetData topic

Returns a list of image fields found in the named Commence category.

The list is of the form:

{Field 1}Delim {Field 2}Delim....Delim{Field n}

where {Field n} is the name of the nth field.

Note: If the optional Delim parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The Delim parameter can be any string (up to 8 characters).

GetImageFieldToFile

Syntax: [GetImageFieldToFile(*Category, Item, Field, Filename*)]

Request Item for the GetData topic

Copies the image field identified by the named *Category*, *Item*, and *Field* to the specified *Filename*. Only bitmap (.BMP) format image fields are currently supported..

Filename must be a fully qualified path including the drive letter (e.g. c:\tmp\data.bmp). If the file does not exist, Commence will create it; the drive and directory, however, must already exist. If the file exists, it will be overwritten. It is the client's responsibility to perform the necessary file cleanup.

See Also:

Parameter Notes

GetItemCount

Syntax: [GetItemCount(*Category*)]

Request Item for the GetData topic

Returns the number of items in the named Commence category.

GetItemNames

Syntax: [GetItemNames(*Category, Delim*)]

Request Item for the GetData topic

Returns a list of all items in the named Commence category.

The list is of the form:

{Item 1}Delim **{Item 2}**Delim**....**Delim**{Item n}**

where **{Item n}** is the name of the nth item.

The amount of data returned by **GetItemNames** may be very large; several kilobytes is very possible.

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

GetLastError

Syntax: [GetLastError]

Request Item for the GetData topic

Returns the error code number from the most recent request or execute error.

This request is now necessary since the switch to DDEML means error codes are no longer sent back with the NACK message. DDE NACK error codes are also displayed in the Commence Message Log.

See Also:

error codes.

GetLetterViewInfo

Syntax: [GetLetterViewInfo(*Delim*)]

Request Item for the GetData topic

Returns information about the active view and selected letter template at the time the Tools-Send Letter or Customize-Database-Letter Template command was executed. This call is used by the Commence letter macros.

The list is of the form:

{ViewName}Delim **{Type}**Delim **{Category}**Delim **{ItemName}**Delim **{FieldLabel}** Delim**{Template Name}**Delim**{Template Filename}**Delim**{Destination}**Delim**{Preview}**Delim**{Rowcount}**Delim**{Letter Filename}**

{Type} can be any one of the following:

Book

Calendar

Gantt Chart

Report

Item Detail

{FieldLabel} is only valid if the active child window is an item detail. Note that a GetField on an active item detail will only retrieve the previous field value (before the edit). Until an item is saved, there is no way to retrieve updated field values.

{Template Name}

Name of the letter template

{Template Filename}

full path and filename to the letter template file; with .DOT extension ?

{Destination}

1 = printer

2 = fax

3 = email

{Preview}

0 = no preview

1 = preview

{Rowcount}

Number of items to merge (1= single letter; > 1 is a mail merge)

{Letter Filename}

full path and filename to the merged document file; this may be an auto-generated name or custom filename; if this is blank, the user has selected not to LINK the document

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

GetMarkItem

Syntax: [GetMarkItem(*Category, Item, Clarify Value*)]

Request Item for the GetData topic

“Marks” the specified item and makes it the default category and item for subsequent calls to:

EditItem, *DeleteItem*, *AppendText*, *AssignConnection*, *UnassignConnection*, *PromoteItemToShared* and *ShowItem* EXECUTE commands, and

GetField, *GetFieldToFile*, *GetFields*, *GetConnectedItemCount* and *GetConnectedItemNames* REQUESTS.

The category and item is marked only for the specific DDE conversation, not globally. that is, each DDE conversation can have a different marked item.

Can be used to mark the (-Me-) item if one is defined.

Syntax: [GetMarkItem(”(-Me-)”)]

Note: If a Clarify Field is defined for the category, the value of the clarify field can be specified with an optional third paramter. Alternatively, the ItemName can be a clarified item name.

See Also:

Parameter Notes

GetMeField

Syntax: [GetMeField(*Field*)]

Request Item for the GetData topic

Returns the data for the specified Field from the (-Me-) item, if one is defined.

GetPhoneNumber

Syntax: [GetPhoneNumber(*PhoneNumber*)]

Request Item for the GetData topic

Returns a phone number. Uses the current TAPI settings to determine the complete phone number to be dialed.

For example, if the local area code is 908 and the dial tone access number is “9,” then passing in “(908)530-4666” will return “9,530-4666”.

TAPI must be configured for GetPhoneNumber to work properly.

GetPreference

Syntax: [GetPreference(*Setting, Delim*)]

Request Item for the GetData topic

Returns some Commence preferences setting (from **Customize-Preferences**). Valid *Settings* are:

"Me"	Returns the Category and Item Name of the selected Me item (delimited by CR/LF). For example, "Person{Delim}Doe,John"
"ExternalDir"	Returns the full path of the Commence external data file directory. For example, "c:\commence\data\files".
"LetterLogDir"	Returns the full path of the Commence letter Log directory. For example, "c:\commence\letters".

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

GetReverseName

Syntax: [GetReverseName(*Name, PrefFlag*)]

Request Item for the GetData topic

Uses Commence preference information and reverses the name if appropriate. The returned string is “Last, First” (i.e., unchanged) if the reverse name check box is not checked, or “First Last” if the check box is checked.

Note: If the optional *PrefFlag* parameter is 1, then the **Customize-Preferences-Other-Name Field** setting is used to determine if names should be reversed If the *PrefFlag* parameter is left blank or is set to 0, then the command uses the Reverse Name checkbox value of the latest invocation of **Tools-Send Letter** and reverses the name if appropriate.

GetTriggerCount

Syntax: [GetTriggerCount()]

Request Item for the GetData topic

Returns the number of agents defined with DDE trigger types.

GetTriggerNames

Syntax: [GetTriggerNames(*Delim*)]

Request Item for the GetData topic

Returns a list of all trigger names that were are defined in agents with the Receive DDE trigger (i.e. the arg value). Since multiple agents can be waiting for the same trigger string, there may be duplicates.

The list is of the form:

{DDETrigger 1}Delim **{DDETrigger 2}**Delim**....**Delim**{DDETrigger n}**

where **{DDETrigger n}** is the name of the nth DDE trigger.

The retrieved trigger names can be used in conjunction with the FireTrigger Execute command to actually fire the agent.

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

GetViewCount

Syntax: [GetViewCount(*Category*)]

Request Item for the GetData topic

Returns the number of defined views for the *Category.*

Note: If the optional parameter *Category* is not specified, the number of defined views for all categories is returned.

GetViewNames

Syntax: [GetViewNames(*Category ,Delim*)]

Request Item for the GetData topic

Returns a list of the defined views for the *Category.*

The list is of the form:

{View 1}Delim **{View 2}**Delim**....**Delim**{View n}**

where **{View n}** is the name of the nth view.

Note: If the optional parameter *Category* is not specified, the list contains the defined views for all categories. If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

GetViewToFile

Syntax: [GetViewToFile (Viewname, Mode, Param1, Param2, Filename)]

Request Item for the GetData topic

Saves the Commence View to a file in HTML format (similar to the File-Save As HTML menu command). If the file does not exist, Commence will create it; the drive and directory, however, must already exist. If the file exists, it will be overwritten. It is the client's responsibility to delete the file. The data is written out in text format, with paragraphs delimited with CR/LF (ASCII codes 13 and 10).

The return string value is "OK" on success.

Mode, Param1, Param2 are used with view linking to identify the "active item/date/date range" when the view contains a "view linked" filter. (example: view of History filtered by the active Account item.)

Mode is one of the following values

0: no view linking

1: view linking on selected item; Param1 = category name; Param2 = Item name [clarify issues?]

2: view linking on selected date; Param1 = date string (including AI dates); Param2 = unused

3: view linking on selected date range; Param1 = start date string (including AI dates); Param2 = end date string (including AI dates)

LogPhoneCall

Syntax: [LogPhoneCall(*Category1, Item1, ..., Category n, Item n*)]

Execute Item for GetData and ViewData topics

Launches an add item detail window for an item in the Phone Log category (as specified in the Commence **Customize-Preferences-Event Logs** dialog box). One or more Category/Item pairs may be given as arguments; connections will be set between the new Phone Log item and these items. If no connection exists between the two categories, the pair is ignored and no error is given. The connection is set via all connections found between the two categories.

If no Phone Log category is set, NACK error code 104 (DDE_ERROR_NOLOGCLASS) is returned. If there is no room to add an item in the Phone Log category, NACK error code 112 (DDE_ERROR_FULL) is returned.

Note: Additional parameter pairs of *Category, Item* may be specified as needed.

See Also:

Execute

Parameter Notes

MarkActiveItem

Syntax: [MarkActiveItem]

Request Item for the GetData topic

“Marks” the highlighted item in the active view and makes it the default category and item for subsequent calls to:

EditItem, *DeleteItem*, *AppendText*, *AssignConnection*, *UnassignConnection*, *PromoteItemToShared* and *ShowItem* EXECUTE commands, and

GetField, *GetFieldToFile*, *GetFields*, *GetConnectedItemCount* and *GetConnectedItemNames* REQUESTS.

MergeTemplateCreate

Syntax: [MergeTemplateCreate (name*, Category, S*hared)]

Request Item for the GetData topic

Creates a new template with the given parameters. if duplicate named template exists, the call fails.

This call returns the template filename assigned by Commence, including the full path and extension configured in Preferences-Letter dialog.

Boolean Shared : 0 | 1

Used to mark the template as shared. Shared is ignored for single-user, non-authors

MergeTemplateSave

Syntax: [MergeTemplateSave (name, Shared)]

Request Item for the GetData topic

Updates an existing template identified by the name parameter. Template must exist or the call fails.

The purpose of this call is to mark the template as changed for synchronization purposes, since the macros can write to the template file directly without involving Commence.

This call returns the template filename assigned by Commence, including the full path and extension configured in Preferences-Letter dialog.

Boolean Shared : 0 | 1

Used to mark the template as shared. Shared is ignored for single-user, non-authors

PromoteItemToShared

Syntax: [PromoteItemToShared(*Category, Item*)]

Execute Item for GetData and ViewData topics

Promotes a local item to shared status, provided the database is connected and the category is shared. If the item is already shared, its status is unchanged. Note that there is no way to demote an item from shared to local status.

Whenever possible, use a clarified *Item* name.

See Also:

Execute

Parameter Notes

ShowDesktop

Syntax: [ShowDesktop(*Desktop Name*)]

Execute Item for GetData and ViewData topics

This command opens the specified desktop in Commence.

ShowItem

Syntax: [ShowItem(*Category, Item*)]

Execute Item for GetData and ViewData topics

Opens an item detail window in Commence for the given *Item* in the given *Category*. If a detail is already open for the item, it will be brought to the top and given the focus.

Note: Starting in version Commence 3.0a, ShowItem can specify the detail form to be used in the optional third parameter. Use the following syntax:

[ShowItem(*Category, Item, FormName*)]

See Also:

Execute

Parameter Notes

ShowView

Syntax: [ShowView(*View Name, 1*)]

Execute Item for GetData and ViewData topics

This command opens the specified view in Commence.

If the view is already open, it will become the active view.

Note: If the view is already open, specify [ShowView(*View Name, 1*)] to force a new copy of the view to be opened.

Status

Syntax: Status

Request Item for the System topic

Returns “Busy” if DDE is unavailable (for example if Commence is currently printing, agents are running, or workgroup syncing is taking place). Returns “Ready” to indicate when Commence is available to process DDE Requests.

SysItems

Syntax: SysItems

Request Item for the System topic

Returns a tab-delimited list of items supported by the System topic.

Topics

Syntax: Topics

Request Item for the System topic

Returns a tab-delimited list of supported DDE conversation topics.

UnassignConnection

Syntax: [UnassignConnection(*FromCategory, FromItem, Connection, ToCategory, ToItem*)]

Execute Item for GetData and ViewData topics

Unassign the connection between the two indicated *Items*. Whenever possible, use a clarified *Item* name.

See Also:

Execute

Parameter Notes

Version

Syntax: Version

Request Item for the System topic

Returns the Commence version number as it appears in the **Help-About** box, in the form "Version *x.y*" where *x* and *y* are integers.

VersionExtended

Syntax: VersionExtended

Request Item for the System topic

Returns the full Commence version number plus the update level in the form "Version *x.y.z.w*" where *x, y, z* and *w* are integers. The full version also appears in the **Help-System Information** box under Version Information.

ViewCategory

Syntax: [ViewCategory(*Category*)]

Request Item for the ViewData topic

Defines the category to be used. This must be the first message sent to Commence for a ViewData conversation. ViewCategory may be sent at any time to reset the ViewData conversation state.

See Also:

ViewView

ViewConjunction

Syntax: [ViewConjunction(*AndOr12, AndOr13, AndOr34*)]

Request Item for the ViewData topic

Defines the logical operations that link the filter clauses specified with the ViewFilter request. Each parameter can be either "And" or "Or"; the default value is "And". *AndOr12* defines the relationship between the filter clauses 1 and 2, *AndOr34* for clauses 3 and 4, and *AndOr13* for (1 and 2) and (3 and 4).

The logical precedence is as follows:

(Filter1 *AndOr12* Filter2) *AndOr13* (Filter3 *AndOr34* Filter4)

The NotFlag defined in the ViewFilter request has, of course, the tightest binding.

ViewConnectedCount

Syntax: [ViewConnectedCount(Index, ConnectionName, ToCategory)]

Request Item for the ViewData topic

Returns the number of items in the *ToCategory* that the *Index*th item is connected to via *ConnectionName*.

ViewConnectedField

Syntax: [ViewConnectedField(*Index, ConnectionName, ToCategory, ConnIndex, Field*)]

Request Item for the ViewData topic

Returns the value of *Field* of the *ConnIndex*th item in the *ToCategory* that is connected to the *Index*th item via *ConnectionName*. Use the ViewConnectedCount REQUEST to determine the maximum *ConnIndex* value. The string "(deleted)" will be returned if the requested item is deleted after the ViewData snapshot is taken.

Field cannot be of the field type Image.

ViewConnectedFields

Syntax: [ViewConnectedFields(*Index, ConnectionName, ToCategory, ConnIndex, n, Field_1..., Field_n, Delim*)]

Request Item for the ViewData topic

Returns multiple field values (*Field_1* through *Field_n)* in the *ConnIndex*th item in the *ToCategory* that is connected to the *Index*th item via *ConnectionName*. Use the ViewConnectedCount REQUEST to determine the maximum *ConnIndex* value. The string "(deleted)" will be returned if the requested item is deleted after the ViewData snapshot is taken.

Field cannot be of the field type Image. The returned data is of the form:

{Field Value 1}Delim **{Field Value 2}**Delim....Delim**{Field Value n}**

where **{Field Value n}** is the value of the nth field in the *ConnIndex*th item.

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

ViewConnectedItem

Syntax: [ViewConnectedItem(*Index, ConnectionName, ToCategory, ConnIndex*)]

Request Item for the ViewData topic

Returns the *ConnIndex*th item in *ToCategory* that the *Index*th item is connected to via *ConnectionName*. Use the ViewConnectedCount REQUEST to determine the maximum *ConnIndex* value.

ViewDeleteAllItems

Syntax: [ViewDeleteAllItems]

Request Item for the ViewData topic

Deletes all items that satisfy the currently defined filter. If no filter is defined, deletes all items in the previously named category (see ViewCategory).

Use with care!!

ViewField

Syntax: [ViewField(*Index, Field*)]

Request Item for the ViewData topic

Returns the value of *Field* of the *Index*th item that satisfies the currently defined filter (if any). Use the ViewItemCount REQUEST to determine the maximum *Index* value. The string "(deleted)" will be returned if the requested item is deleted after the ViewData snapshot is taken.

FieldName cannot be of the field type Image.

Note: If the Name field is requested, the returned value is NOT clarified. Use ViewItemNameand ClarifyItemNames) to retrieve the clarified item name.

ViewFields

Syntax: [ViewFields(*Index, n, Field_1, Field_2 ,..., Field_n, Delim*)]

Request Item for the ViewData topic

Returns multiple field values (*n*) for the *Index*th item that satisfies the currently defined filter (if any). Use the ViewItemCount request to determine that maximum *Index* value. Use the GetFieldNames request to determine valid field names.

The string "(deleted)" will be returned if the requested item is deleted after the ViewData snapshot is taken. Image type fields are not allowed in the *Field_* parameters.

The returned data is of the form:

{Field 1}Delim **{Field 2}**Delim....Delim**{Field n}**

where **{Field n}** is the name of the nth field name.

Note: If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

ViewFieldToFile

Syntax: [ViewFieldToFile(*Index,Field,Filename*)]

Request Item for the ViewData topic

Identical to ViewField except the data is copied to the specified *Filename*. The data is written out in text format, with paragraphs delimited with CR/LF (ASCII codes 13 and 10).

Filename must be a fully qualified path including the drive letter (e.g. c:\tmp\data.txt). If the file does not exist, Commence will create it; the drive and directory, however, must already exist. If the file exists, it will be overwritten. It is the client's responsibility to perform the necessary file cleanup.

ViewFilter

Syntax: [ViewFilter(*ClauseNumber, FilterType, NotFlag, FilterTypeParameters*)]

Request Item for the ViewData topic

Defines the criteria for the multiple filter to be applied against the previously named category (see ViewCategory).

ClauseNumber defines which filter clause is being defined, where *ClauseNumber* is between 1 and 4.

FilterType sets the type of the filter to apply.

NotFlag determines if a logical Not is applied against the entire clause. The *NotFlag* parameter must be specified; it may be either **Not** or left blank (by specifying the comma placeholder for that parameter). This is equivalent to the “Except” checkbox found in the Commence filter dialog boxes.

The *FilterTypeParameters* specified with this REQUEST and the ViewConjunction REQUEST are similar to those available from the Commence main menu.

View Filter Examples

Valid Filter Types for ViewFilter

ViewFilter will accept the following abbreviations for *FilterType*. The spelled out filter type or the abbreviation may be specified.

FilterType	***Abbreviation***
Field	*F*
Connection To Item	*CTI*
Connection To Category To Item	*CTCTI*
Connection To Category Field	*CTCF*

Valid Parameters for ViewFilter

The *FilterType* determines what additional parameters must follow the *NotFlag* field. The parameters are summarized in the following table:

FilterType	***Filter Type Parameters***
Field	*FieldName, Qualifier, Value, CSFlag*
Connection To Item	*ConnectionName, ConnectedCategory, ItemName*
Connection To Category To Item	*ConnectionName1, ConnectedCategory1, ConnectionName2, ConnectedCategory2, ItemName*
Connection To Category Field	*ConnectionName, ConnectedCategory, FieldName, Qualifier, Value, CSFlag*

Filtering by a Field

FieldName defines the field to be filtered.

The Qualifier depends on the field type.

Value specifies the value to compare the field against.

CSFlag determines if the comparison is case-sensitive or not. *CSFlag* is only necessary for textual comparisons. Values of **"1"**, **"yes"**, or **"true"** will force a case-sensitive comparison, while **"0"**, **"no"**, or **"false"** will force a case-insensitive comparison.

Note: The *Value* and *CSFlag* fields are not relevant for certain field qualifications. For "Is Blank" filtering and for Check Box fields, the *Value* and *CSFlag* fields can be left blank or omitted.

When filtering for a range of values, the *Value* and *CSFlag* fields are replaced by *Min Value* and *Max Value*. Range filtering always does an inclusive comparison with the specified minimum and maximum values.

Clearing a Filter

The ViewFilter REQUEST can clear a filter. To do so, specify the filter to be cleared (1, 2, 3, or 4) and include the word CLEAR. For example, issuing [ViewFilter(1,Clear)] will clear filter 1.

Filtering by Shared vs. Local status

The ViewFilter REQUEST can filter for shared or local items. To do so, see **Filtering by a Field** above. Specify either "Shared" or "Local" for the *Qualifier*. Leave the *FieldName*, *Value*, and *CSFlag* parameters empty). For example,

```
[ViewFilter(1, F, , , Shared, , )]
[ViewFilter(2, CTCF, , Is Employed by, Company, , Local)]
```

Filtering by a Connection

The *ConnectionName*, *ConnectedCategory* parameter pair defines the connection from the previously selected (base) category to a different Commence category. ItemName is the name of the item in the connected category.

View Filter Examples

Valid Qualifiers for ViewFilter

The ***FieldTypeParameters*** (e.g. *FieldName/Qualifier/Value/CSFlag*) define a field and the value of the field to filter by. The *Qualifier* depends on the field type of *FieldName*. The following chart defines which Qualifiers are valid for the various Commence field types:

Field Type	***Valid Qualifiers***
Calculation	Equal To, Not Equal To, Less Than, Greater Than, Between
Check Box	1, 0, Checked, Not Checked, True, False, Yes, No,
Date	Before, On, After, Between, Blank
E-Mail Address	Contains, Doesn't Contain, Equal To, Between, Blank
Internet Address	Contains, Doesn't Contain, Equal To, Between, Blank
Name	Contains, Doesn't Contain, Equal To, Between, Blank
Number	Equal To, Not Equal To, Less Than, Greater Than, Between
Selection	Equal To, Not Equal To
Sequence Number	Equal To, Not Equal To, Less Than, Greater Than, Between
Telephone	Contains, Doesn't Contain, Equal To, Blank
Text	Contains, Doesn't Contain, Equal To, Between, Blank
Time	Before, At, After, Between, Blank
N/A	Local, Shared

Data file fields, Excel cell fields, and image fields cannot be used in a filter.

ViewFilter Examples

An example of how to create filters:

```
[ViewCategory("Person")]
[ViewFilter("1","F",,"Address", "Contains", "NJ", "yes")]
[ViewFilter("2","CTI",,"Is Employed by", "Company", "Great Homes Inc.")]
[ViewFilter("3","CTCTI",,"Is Employed by", "Company", "Associated with", "Project", "Storage
        Wizard")]
[ViewFilter("4","CTCF", ,"Is Employed by", "Company", "City", "Contains", "Seattle", "yes")]
[ViewItemCount()]
```

ViewFilter is by far the most complicated DDE request. Here are some miscellaneous examples to demonstrate how powerful it is.

`Request: [ViewFilter(1, F,, Title, Doesn't Contain, ?,)]`

Filter for all items with a blank Title field.

`Request: [ViewFilter(1, F, Not, Title, Contains, ?,)]`

Equivalent to previous example.

`Request: [ViewFilter(1, F,, Title, Blank)]`

Equivalent to previous example.

`Request: [ViewFilter(1, CTI,, Is Employed by, Company, "Appliance Distributors Inc")]`

Filter for all people employed by Appliance Distributors Inc.

`Request: [ViewFilter(1, CTCF,, Is Employed by, Company, State, Contains, NJ, yes)]`

Filter for all people employed by Companies in the state of New Jersey (State contains the case-sensitive text "NJ")

`Request: [ViewFilter(1, CTCTI,, Attends, Appointment, Relates to, Project, Space Layouts)]`

Filter for all people attending any appointment concerning the Space Layouts project.

`Request: [ViewFilter(1, CTCF,, Attends, Appointment, Date, Between, 1/1/95, today)]`

Filter for all people attending appointments from January 1, 1995 to today.

DDE Examples

ViewImageFieldToFile

Syntax: [ViewImageFieldToFile(*Index,Field,Filename*)]

Request Item for the ViewData topic

Copies the image field data in *Field* from the *Index*th item that satisfies the currently defined filter (if any) to the specified *Filename*.

Filename must be a fully qualified path including the drive letter (e.g. c:\tmp\data.bmp). If the file does not exist, Commence will create it; the drive and directory, however, must already exist. If the file exists, it will be overwritten. It is the client's responsibility to perform the necessary file cleanup.

ViewItemCount

Syntax: [ViewItemCount()]

Request Item for the ViewData topic

Returns the number of items that satisfy the currently defined filter (if any). If no filter is defined, this request returns the total number of items in the category.

ViewItemIndex

Syntax: [ViewItemIndex(*NameFieldValue*)]

Request Item for the ViewData topic

Returns the index of the item with the indicated name field value. The item must also satisfy the currently defined filter (if any).

Note: If the *NameFieldValue* is left blank, the marked item will be used. Returns the index of the marked item (if any).

ViewItemName

Syntax: [ViewItemName(*Index*)]

Request Item for the ViewData topic

Returns the item name of the *Index*th item that satisfies the currently defined filter (if any). Different from ViewField(*Index, Name*) in that a clarified item name may be returned. The ClarifyItemNames request is used to enable or disable the use of clarified item names.

ViewMarkItem

Syntax: [ViewMarkItem(*Index*)]

Request Item for the ViewData topic

Marks the *Index*th item in the view and makes it the default category and item for the EditItem, AppendText, AssignConnection, UnassignConnection, PromoteItemToShared and ShowItem EXECUTE commands.

ViewReverseName

Syntax: [ViewReverseName(*Name, PrefFlag*)]

Request Item for the ViewData topic

Uses the **Customize-Preferences-Other-Name Field** setting and reverses the name if appropriate. The returned string is “Last, First” (i.e., unchanged) if the check box is not checked, and is “First Last” if the check box is checked.

Note: If the optional *PrefFlag* parameter is 1, then the **Customize-Preferences-Other-Name Field** setting is used to determine if names should be reversed If the *PrefFlag* parameter is left blank or is set to 0, then the command uses the Reverse Name checkbox value of the latest invocation of **Tools-Send Letter** and reverses the name if appropriate.

ViewSaveView

Syntax: [ViewSaveView(*New View Name, Shared*)]

Request Item for the ViewData topic

Saves the virtual DDE view already created with the ViewCategory, ViewFilter, ViewSort, etc., as an actual view in Commence, accessible via the normal user interface. The view will be saved using the specified *New View Name*.

CAUTION: If the view already exists, it will be overwritten. Use GetViewNames to determine if the view name conflicts with an existing view.

Note: In a workgroup situation, the optional *Shared* parameter determines whether the view is local or shared. Valid values are "**yes**" and "**no**"

ViewSort

Syntax: [ViewSort(*Field1, Sort1, Field2, Sort2, Field3, Sort3, Field4, Sort4*)]

Request Item for the ViewData topic

Defines the sort criteria for the view. *Field1, Field2, Field3,* and *Field4* define the fields to sort by. Sort1, Sort2, Sort3, and Sort4 define the sort type and should be "**Ascending**" or "**Descending**". Unused sort pairs (i.e. FieldN, SortN) may be omitted.

ViewView

Syntax: [ViewView(*ViewName*)]

Request Item for the ViewData topic

Uses the category and filter criteria of a Commence View that has already be defined from the Commence menu interface. The **ViewView** REQUEST overrides any previously received ViewCategory, ViewConjunction, ViewFilter, and ViewSort REQUESTS.

If the *ViewName* parameter is left blank, the active view will be used.

Notes

Dynamic Data Exchange Overview
Miscellaneous DDE Notes
Miscellaneous Topic Notes
Miscellaneous ViewData notes
Miscellaneous Execute notes
DDE Pokes
Managing Multiple Conversations
Implementation Notes
Clarified Item Names
Item Marking
Value Passing from the Agent Receive DDE Trigger

Dynamic Data Exchange Overview

Dynamic Data Exchange (DDE) is a form of communication that allows Windows applications to communicate with each other. This communication is referred to as a DDE *conversation*. The application which initiates the conversation is known as the *client*. The application responding to the client is known as the *server*. The client can request data from the server, send data to the server, and perform any other tasks which are allowed by the server. Commence can function as both a client and a server, which means Commence can initiate a DDE conversation with any DDE capable application, including itself.

Multiple DDE conversations can occur simultaneously, where the application may be acting as a server for some conversations and a client in others. To keep the DDE messages straight, each conversation must occur on its own *channel*. The client application opens a DDE channel by sending an Initiate command to the server application. During the conversation, the client may perform any number of functions which can be broken down into three types: *Requests*, *Pokes*, and *Executes*. Requests are typically used to get data from the server, Pokes are used to send data to the server, and Executes send commands to be executed by the server. When the client is finished processing all the DDE commands, it is the client's responsibility to send a Terminate command to the server to close the DDE channel. See Managing Multiple Conversations for more information.

Implementing DDE requires knowledge of and access to a macro language (e.g., MS Word for Windows, MS Excel, Lotus Ami Pro) or an application development tool (e.g., MS Windows SDK, Visual Basic, Turbo Pascal for Windows). A macro language or other programming tool allows you to organize many DDE commands to be performed in the same DDE conversation. The Commence letter macros are a common example of how to use a macro language (e.g. , Word Basic) to communicate with Commence via DDE. For example, the Word 6.0 file called JJMACS6.DOT in the winword\template directory contains a macro called "Commence Sample", which illustrates some basic techniques for sending DDE messages to Commence. Commence does not have it's own macro language, so you must initiate all your DDE conversations from an Agent. Typically, you could either use the Application Launch action to launch a program or macro written by another application, such as Visual Basic, or you could use the Send DDE action and/or Receive DDE trigger.

When using Commence Agents to send DDE messages to another application or to itself, Commence performs all the necessary DDE Initiates and Terminates for you. All you need to specify is the actual DDE command, which can be either an Execute or a Poke. (Note: Commence agents cannot perform DDE Requests; but Requests can be made from another client application where Commence is acting as the DDE Server.) You will also have to identify the following things:

> **Program Filename**. Enter the full path to the executable for the server application (i.e. the application to which Commence is sending DDE messages). For example, if Commence is the server application, the Program Filename would be "c:\commence\commence.exe".
>
> **Application Name**. Enter the DDE Application name for the server application. This is usually the program name without the .EXE extension, but check the manual for the other application to be sure. The application name for any version of Commence is always "Commence".
>
> **Topic Name**: Enter a valid topic name. See Miscellaneous Topic Notes for a list of valid topics for Commence. For applications that use file-based documents, such as Excel, the topic name is usually the filename. If it is not clear what the topic is for another application, then use "System" which is accepted by all DDE capable applications.

See the Commence On-Line Help (commence.hlp) for more detailed information about the DDE Agent triggers and actions. The sample agents in the tutorial database called "Send DDE" and "Receive DDE" illustrate how to use Agents to send DDE messages from Commence to itself.

See Also:

Value Passing from the Agent Receive DDE Trigger

Miscellaneous DDE Notes

DDE hot-links (*Advise*, *Unadvise*) are not supported by Commence. All Commence DDE conversations are cold-links; that is, if a client application retrieves some data from Commence and that data is subsequently updated, it is the client application's responsibility to retrieve the new data value. No notification is sent from Commence.

Data from all field types can be exchanged via DDE with the following caveats:

- The value returned for a *Data File* field is the path name of the file.
- The value returned for an *Excel Cell* field is the name of the cell (in <sheetname>!<cellname> format) and not the actual cell value. Getting the actual cell value would require DDE to Excel and this would be quite slow.
- *Sequence Number* fields can be retrieved from Commence but not edited.

Names of categories, fields, connections, field values, etc. must be passed to Commence exactly as they were originally entered. This means that all parameters are case-sensitive and that leading and trailing blanks are significant.

The double quotes surrounding parameters for REQUEST items and EXECUTE commands are optional unless there are imbedded leading or trailing spaces, commas, right parentheses or double quotes.

The following examples are equivalent:

```
[AddItem("Person","John Doe")]
[AddItem(Person,John Doe)]
```

The following examples are not equivalent:

```
[AddItem("Person","Doe, John")]
[AddItem(Person,Doe, John)]
```

(3 parameters)

```
[AddItem(Person," Doe, John")]
```

(leading space, 2 parameters)

To embed a double quote in a parameter, use two double quotes. For example:

```
[AddItem(Person,"Paul ""P.B."" Benson")]
```

adds Paul "P.B." Bensen to the Person category.

Commence can support multiple simultaneous DDE conversations. For example, another application can open both a GetData and multiple ViewData conversations in the same session.

~~See Also:~~

Miscellaneous Execute Notes

Notes

Miscellaneous Topic Notes

Commence supports the following Dynamic Data Exchange conversation topics:

System
GetData
ViewData
MergeItem
The name of the currently active database (e.g. Main Database)
The full path of the currently active database (e.g. c:\commence\data)

The System topic retrieves some basic DDE status information. The GetData topic is used for one-time data transfers. The ViewData topic is used to create a snapshot of a Commence category and then to retrieve various fields from that snapshot. The snapshot is analogous to a report view of a Commence category. Data can then be retrieved from Commence based on the view. The MergeItem topic automatically marks the active item in Commence, enabling later DDE calls to get field values from the active item.
The GetData and ViewData topics are interchangeable. Get* and View* requests can be used with either topic. Requests can be intermixed on the same conversation. GetData and View Data are included for compatibility with Commence 1.x and 2.x. These topics may not be supported in the future. The recommended topic conversation with Commence is the full name or the path of the currently active database.

By using the database name or path as the conversation topic, multiple instances of Commence can peacefully co-exist on the same machine. DDE clients are also assured that they are conversing with the expected instance of Commence.

Since DDE conversations maintain state information that is dependent on the currently active database, Commence will terminate active DDE conversations if the user opens a new database (from **File-Open/Manage Database**) or if tutorial mode is entered (**Help-Tutorial**). The System Topic is the only exception; System conversations will stay up regardless of any database changes.

Miscellaneous ViewData Notes

A **ViewData** conversation maintains a snapshot of the viewed category. This snapshot is taken with the first ViewItemCount, ViewField, ViewConnectedCount or ViewConnectedItem REQUEST. Subsequent ViewCategory, ViewFilter or ViewView REQUESTs invalidate the snapshot.

ViewData REQUESTs that complete successfully return either the requested information, if applicable, or the string "OK". REQUESTs that fail return NACKS.

If a view is filtered by connections, that filter may be invalidated if the connected category is modified (either via DDE or the Commence menu interface).

Changing the Commence database and entering/leaving Tutorial mode from the Commence menu interface changes the underlying set of available categories. During a **ViewData** conversation, of the Commence database directory is changed or if the Tutorial mode is entered/left, then the view is invalidated.

~~See Also:~~

NACK Error Codes

Miscellaneous DDE Notes

Miscellaneous ViewData Notes

Notes

Miscellaneous Execute Notes

Executes that complete successfully return an ACK. Executes that fail return NACKs.

Multiple commands per DDE message are supported for Execute (but not for Request). Multiple commands should be of the form: [command 1][command 2]...[command n].

Commence category and connection definitions cannot be added, edited or deleted via DDE.

~~See Also:~~

NACK Error Codes

Miscellaneous DDE Notes

Miscellaneous ViewData Notes

Notes

DDE Pokes

Pokes are used by the client application to send data to the server application. A Commence Agent can perform a DDE Poke with a Send DDE action. With every Poke, the item name must be specified and the data to be sent must be included in the Poke. For example, to Poke a value from Commence into an Excel cell:

Program Filename. Enter the full path to the executable for the server application (e.g. “c:\msoffice\excel\excel.exe”).

Application Name. Enter the DDE Application name for the server application (e.g. “Excel”).

Topic Name: Enter the filename of the spreadsheet (e.g. “c:\msoffice\excel\foo.xls”).

Item Name: Specify the row/column value of the cell (e.g. “R1C3”).

Message Text: Specify the value (e.g. “$395.00”) or the field code of the Commence field containing the data (e.g. “(%Total Due%)”)

Multiple values can be poked in the same DDE conversation. Enter the list of item names delimited with the piping symbol “|” in the Item Name box, and the list of values delimited with the piping symbol “|” in the Message Text box.

Similar to the above example, a DDE Poke can be used to send a fax from a Commence Agent using WinFax 4.1 or later. See your WinFax documentation for the syntax of the DDE Poke command to use.

Managing Multiple Conversations and Instances

Commence allows multiple instances of itself to be running. That is, two copies of Commence can be running at the same time, opened to different database.

You can specify which instance of Commence to communicate with in two different ways:

By using “CommenceXXXX” as a valid DDE Application name, where XXXX is the Windows instance identifier. See MS Knowledgebase article Q105659 for information on how to determine XXXX and to see how this is useful.

By using the active database name or path as the DDE Topic. For example:

```
DDEInitiate(“Commence”,”Main Database”)
DDEInitiate(“Commence”,”c:\commence\data”)
DDEInitiate(“Commence1234”,”GetData”)
DDEInitiate(“Commence1234”,”Main Database”)
```

Whever the Category and item names can be left blank to denote the ‘marked’ item, “(-Me-)” can now be used to refer to the Me item. For example:

```
Execute:    [AssignConnection(Company, JJI, Employs, “(-Me-)”,  “(-Me-)”)]
Execute:    [UnassignConnection(, , Employs, “(-Me-)”,  “(-Me-)”)]
Execute:    [EditItem(“(-Me-)”, “(-Me-)”, Snapshot, c:\image\me.bmp)]
```

Implementation Notes

Commence DDE is implemented using the DDE Management Library (DDEML) supplied by Microsoft. This requires that the DDEML.DLL file be properly installed in the Windows system directory before Commence will boot. (Note: DDEML requires that Windows be running in protected-mode, but for Windows 3.1 or higher, this is not a problem.)

DDEML does impose one difference. In Commence 2.x and earlier versions, every NACK message included an error code that was useful for development and testing. In DDEML, this error code is no longer supported. Instead, Commence will now report DDE errors via **Help-Message Log** and the new GetLastError request.

~~See Also:~~

NACK error codes

Clarified Item Names

In Commence, the maximum size of the item name field is 50 characters. Commence DDE REQUESTS and EXECUTES can accept item names in the following formats:

{ItemName}

{ItemName padded with spaces to 50 characters}{clarify field value}

{ItemName padded with spaces to 50 characters}{clarify field separator}{clarify field value}

For convenience, clarified item names can also use the tab character (0x09) as a delimiter after the item name instead of spaces. In this case, *{clarify field value}* may be the null string.

{ItemName}{TAB}{clarify field value}

{ItemName}{TAB}{clarify field separator}{clarify field value}

For example:

```
[EditItem(Person, "Romanov    - (732)974-9838")]
```

Existing DDE applications that use clarified item names, but only pass in item names that were returned by Commence, should run without modification (since Commence returns values using the longer item name). DDE applications that explicitly build clarified item names will need to be modified to use the longer length or the tab delimiter.

The clarify field separator is optional but highly recommended. For example:

```
[GetField("Appointment", "Lunch                                  - 10/20/92", "Start Time")]
[EditItem(Person, "Doe, John                              - 111-22-3333")]
```

When adding an item via DDE, a clarify field value can be specified in the AddItem command as in the following example:

```
[AddItem(Person, "Doe, John", 111-22-3333)]
```

When retrieving item names, Commence can return both clarified and non-clarified item names. Use the ClarifyItemNames REQUEST to control the item name format. Use GetCategoryDefinition to determine the clarify field separator and clarify field name.

Item Marking

The EditItem, AppendText, AssignConnection and UnassignConnection Execute commands are problematic since they require unique item identification. Even with clarified item names, this is not always possible, To address this problem, Commence supports "item marking", which is nothing more than recording a default category and item.

If the category name and item name parameters are left blank, Commence automatically uses whatever item was last marked. Items can be marked by using the ViewMarkItem Request. Also, any item added via AddItem is automatically marked. The following examples demonstrate how marking can be used:

Example 1:

```
[AddItem(Person, "Doe, John")]          this item automatically marked
[EditItem(,,SSN, 111-22-3333)]
[EditItem(,,Phone, "(123)456-7890")]
[EditItem(,,Title, "VP Marketing")]
[AssignConnection(,, Employed by, Company, Anchorage Ice Company)]
```

Example 2:

```
[ViewCategory(Person)]
[ViewFilter(1,F,,Name, Equal, "Doe, John", CS)]
[ViewFilter(2,F,,SSN, Equal, 111-22-333, CS)]
[ViewItemCount]                         to make sure there is at least one
[ViewMarkItem(1)]                       first item in view is now marked
[EditItem(,,Phone, "(123)456-7890")]
```

Only the EditItem, ShowItem, AppendText, AssignConnection and UnassignConnection Execute commands can use marked items.

Value Passing from the Agent Receive DDE Trigger

Commence can pass data from the Send DDE Action of one agent to the Receive DDE Trigger of another agent. The data can then be used by the Receive DDE agent's actions. The arguments passed via a Receive DDE trigger are accessible using the following keyword strings: `(-0-)`, `(-1-)`, `(-2-)`, `(-3-)`, `(-4-)`, `(-5-)`, `(-6-)`, `(-7-)`, `(-8-)` and `(-9-)`. These keywords (i.e. arguments) may be used just as field codes would be used to display the data in a message box or status bar action, to set values in an add or edit item action, or any other place that keyword replacement is allowed in agents. The following example illustrates how the DDE trigger arguments are mapped to keywords.

Usage:

`[FireTrigger(TrigStr, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7, Arg8, Arg9)]`

(-0-) - This is always FireTrigger.

(-1-) - Trigger String, e.g., TrigStr

(-2-) through (-9-) - The string passed in that position, e.g. Arg2 through Arg9 respectively.

The Send DDE trigger also allows arguments 2 and 3 to be mapped to a category name and an item name respectively. Then If the "**arg2 equals**" check box is checked (in the **DDE Trigger Definition** dialog box of the Receive DDE agent), a category must be selected and `Arg2` must equal that category name for that DDE trigger to fire. In addition, `Arg3` must be the name of an item in the chosen category. If all these conditions are met, the trigger fires and values from that item will be available for use in the agent's actions (using the field name keywords such as `(%Name%)` or using any of the arguments `(-3-)` through `(-9-)` that have been defined)

The corresponding agent with a Receive DDE trigger also allows you to define a filter for a DDE trigger using the item mapping option. If a filter is defined, not only do `Arg2` and `Arg3` have to match the category and an item name, but the item must also meet the filter criteria. If the item does not match the filter, the trigger will not fire.

Examples:

`Execute: [FireTrigger("test", "Person", "Wilson", "908-552-0466")]`

This is a generic example where all arguments are typed literally.

`Execute: [FireTrigger("test",(-Category-),(%Last Name%),(%Business Phone%))]`

This is an equivalent example where the agent trigger uses the Person category, so field codes can be used as arguments whenever necessary.

Keywords used for Value Passing from the Agent Trigger

These keywords can be used in agent conditions whenever it is necessary to refer to fields from the category specified in the trigger. If you define an agent that has trigger data, then the category specified in the condition can be filtered using the keywords described below.

These keywords can also be used in agent actions using values from condition 1 or condition 2, which may also need to reference fields from the trigger item. In cases where an **Insert Field** button is available, these keywords will not be available from the drop-down list. All of the following keywords must be typed into the **Value** text box manually and case-sensitively.

Note: If the condition uses the same category as the agent trigger, then there is no need to use these special trigger keywords; regular field codes can be used instead. The same is true for agent actions using values from trigger. Simply select the **Insert Field** button from the Agent action when available or manually type the field code into the **Value** text box.

(%Trig%%Field Code%)

This keyword passes the field value from the field specified to an agent's condition or action. The field value will be retrieved from the trigger item.

Example where the trigger is a Save Item in the To-Do category:

(%Trig%%Due Date%)

evaluates to the date from the due date field of the To-Do item

(%Trig%%Name%)

evaluates to the name of the To-Do item that was saved

(%Trig%%Connection Name%%Connected Category%)

This keyword passes the field value from the name field of the connected category to an agent's condition or action. This allows you to pass the value of the name field from any item connected to the the trigger item.

Examples where the trigger is a Tool Bar Pick for the Person category:

(%Trig%%Attends%%Appointment%)

evaluates to the names of the appointments for the highlighted person

(%Trig%%Refers to%%Phone Log%)

evaluates to the names of the phone calls for the highlighted person

(-TrigItemName-)

This keyword translates to the name of the item associated with the agent's trigger. If there is no such item, the keyword translates to a zero length string. It is equivalent to the field code representing the Name field from the trigger item, such as (%Trig%%Name%).

Keywords used for Value Passing from the Agent Condition

These keywords can be used in agent actions whenever it is necessary to refer to fields from the category specified in the condition. If you define an agent that has condition data, then a category specified in the action can be filtered using the keywords described below.

Note: Condition keywords can only be used in actions. They cannot be used in the trigger or in another condition. For example, if there is a Condition 2 defined, you cannot use a Cond1 keyword anywhere in the value boxes of the Condition 2.

(%Cond1%%Field%)

This keyword passes the field value from the field specified to an agent's condition or action. The field value will be retrieved from the trigger item. Use "Cond2" to access Condition 2 data.

Example where Condition 1 is an Item Count in the Task category. The action can use:

(%Cond1%%Status%)

evaluates to the value of the Status field for the trigger item

(%Cond1%%Connection Name%%ToCategory%)

This keyword passes the field value from the name field of the connected category to an agent's action. This allows you to pass the value of the name field from any item connected to the the condition item. Use "Cond2" to access Condition 2 data.

Example where Condition 2 is an Item Count in the Task category. The action can use:

(%Cond2%%Assigned to%%Person%)

evaluates to the name of a person

Examples

Commence DDE Examples

This topic provides some examples of how to use DDE with Commence. The reader is assumed to be familiar with DDE concepts. See Dynamic Data Exchange Overview for more information.

In the following examples, `Request:` and `Execute:` are the actual strings that Commence should receive. How the strings are created and sent to Commence are application-dependent and outside the scope of this topic. `Initiate:` specifies the application and topic names for the DDE conversation.

For most character string parameters, Commence does not require double-quote delimiters. Strings need to be double-quote delimited only if they have leading or trailing blanks, or if there is an embedded comma or right parenthesis. For readability, all non-required double-quotes have been left out.

Error handling left as a reader exercise...

Example 1	Filter for a particular person in the Person category. If the person exists, retrieve some field and connection values.
Example 2	Locate To-Do items that are past due and assign new due dates.
Example 3	Red Bank Appliance/Remodeling has gone out of business. All employees will be picked up by parent company, Appliance Distributors Inc. Modify employee "Is Employed by" information.
Example 4	In a workgroup situation, review all local Appointment items and possibly promote them to shared status.
Example 5	You have a phone number and want to use Commence's phone configuration information to determine the exact dial string to be sent to your PBX or modem.
Example 6	Your application's macro language imposes a character string limit so you cannot retrieve a large text field from Commence.
Example 7	Your application's macro language is brain-dead and imposes a limit of 80 characters on the DDE string that it can send. Since GetData requests can easily exceed this size, you can reduce the length of the DDE string by marking the item. Note that the category and item name parameters are left blank in the following examples, so the marked item will be used.
ViewFilter Example	ViewFilter is by far the most complicated DDE request. Here are some miscellaneous examples to demonstrate how powerful it is.

DDE Examples

Update History
02/14/94 added Commence 2.0 and 2.1 information - EPH
05/08/95 added Commence 3.0 information - FV

Example 1

Filter for a particular person in the Person category. If the person exists, retrieve some field and connection values.

```
Initiate: Commence,ViewData
Request: [ViewCategory(Person)]
```

Work with the Person category.

```
Request: [ViewFilter(1, F,,Last Name, Equal to, "Frost",)]
```

Filter for person with name "Frost"

```
Request: [ViewItemCount]
```

Was this person found?

If returns 0, person does not exist in category.

If returns 1, person found.

If returns > 1, multiple people with same name found.

Specify additional filter criteria to get the correct person.

```
Request: [ViewField(1, Street)]
```

Retrieve the person's street address.

```
Request: [ViewField(1, Business Phone)]
```

Retrieve the person's phone number.

```
Request: [ViewConnectedCount(1, Is Employed by, Company)]
```

Check if the person is employed.

If returns 0, no connections to Company exist.

```
Request: [ViewConnectedItem(1, Is Employed by, Company, 1)]
```

Retrieve value of the first connected item. To retrieve other connected items, specify different values for the fourth parameter (can be in range from 1 to [ViewConnectedCount] value).

```
Terminate:
```

DDE Examples

Example 2

Locate To-Do items that are past due and assign new due dates.

```
Initiate: Commence, ViewData
Request: [ViewCategory(To-Do)]
```

Work with the To-Do category.

```
Request: [ViewFilter(1, F,, Completed, No)]
```

Filter clause 1: Completed check box not checked.

```
Request: [ViewFilter(2, F,, Due Date, Before, tomorrow)]
```

Filter clause 2: Due Date of today or earlier.

```
Request: [ViewConjunction(And, And, And)]
```

Make sure it's clause 1 AND clause 2. Not really necessary since this is the default, but it's good form.

```
Request: [ViewItemCount]
```

Find out how many To-do items are past due.

If returns 0, no To-dos are past due.

```
Request: [ViewSort(Priority, Ascending, Due Date, Ascending)]
```

Sort by ascending Priority levels and ascending Due Dates.

```
Request: [ViewField(1, Name)]
```

Retrieve the first To-do. To retrieve other To-dos, specify different values for the first parameter (can be in range from 1 to [ViewItemCount] value).

```
Execute: [EditItem(To-do, <item name>, Due Date, <new date>)]
```

Based on some algorithm or direct user input, re-assign due dates for each item.

```
Terminate:
```

DDE Examples

Example 3

Red Bank Appliance/Remodeling has gone out of business. All employees will be picked up by parent company, Appliance Distributors Inc. Modify employee "Is Employed by" information.

```
Initiate: Commence, ViewData
Request: [ViewCategory(Person)]
```

Work with the Person category.

```
Request: [ViewFilter(1, CTI,, Is Employed by, Company, "Red Bank Appliance/Remodeling")]
```

"Connection to item" filter for all persons employed by Red Bank Appliance/Remodeling.

```
Request: [ViewItemCount]
```

How many people work for Red Bank Appliance/Remodeling?

```
Request: [ViewField(1, Last Name)]
```

Retrieve the first person's name. To retrieve other people, specify different values for the first parameter (can be in range from 1 to [ViewItemCount] value).

```
Execute: [UnassignConnection(Person, <name>, Is Employed by, Company, "Red Bank
      Appliance/Remodeling")]
```

Break connection between person and company.

```
Execute: [AssignConnection(Person, <name>, Is Employed by, Company, "Appliance Distributors
      Inc")]
```

Assign new connection between person and company.

```
Execute: [DeleteItem(Company, "Red Bank Appliance/Remodeling")]
```

Delete the old company item.

```
Terminate:
```

DDE Examples

Example 4

In a workgroup situation, review all local Appointment items and possibly promote them to shared status.

```
Initiate: Commence, ViewData
Request: [ClarifyItemNames(TRUE)]
```

Use clarified item names to reduce ambiguity due to items with identical names.

```
Request: [ViewCategory(Appointment)]
```

Work with the Appointment category.

```
Request: [ViewFilter(1, F, , , Local)]
```

Filter clause 1: Only consider local items

Note: For a standalone Commence database, all items are considered local.

```
Request: [ViewItemCount]
```

Find out how many Appointment items are local.

```
Request: [ViewItemName(1)]
```

Get the (possibly clarified) name of the item.

To retrieve other Appointment, specify different values for the first parameter (can be in range from 1 to [ViewItemCount] value).

```
Request: [ViewFields(1, 3, Date, Start Time, End Time)]
```

Get the Date, Start Time and End Time field values.

To retrieve other Appointment, specify different values for the first parameter (can be in range from 1 to [ViewItemCount] value).

```
Request: [ViewMarkItem(1)]
```

'Mark' the item as the default category and item.

```
Execute: [PromoteItemToShared( , )]
```

Based on some algorithm or direct user input, promote the item from local to shared status. Since the category and item names are left blank, the 'marked' item will be used.

```
Terminate:
```

DDE Examples

Example 5

You have a phone number and want to use TAPI's phone configuration information to determine the exact dial string to be sent to your dialing device (e.g. modem).

```
Initiate: Commence, GetData
Request: [GetPhoneNumber("(908)555-1212")]
```

Commence will respond with something like "9,1 (908)555-1212" (if long-distance) or "555-1212" (if local).

```
Terminate:
```

DDE Examples

Example 6

Your application's macro language imposes a character string limit so you cannot retrieve a large text field from Commence.

```
Initiate: Commence, GetData
Request: [GetFieldToFile(Project, "Freezer Repair Training", Notes, "c:\tmp\notes.txt")]
```

Copy the field contents to a file (which can then be processed by the word processor).

```
Terminate:
```

DDE Examples

Example 7

Your application's macro language is brain-dead and imposes a limit of 80 characters on the DDE string that it can send. Since GetData requests can easily exceed this size, you can reduce the length of the DDE string by marking the item. Note that the category and item name parameters are left blank in the following examples, so the marked item will be used.

```
Initiate: Commence, GetData
Request: [GetMarkItem(Company, "Red Bank Appliance/Remodeling")]
```

'Mark' the item as the default category and item.

```
Request: [GetFields( , , 3, City, State, Zip)]
```

Get some field values.

```
Request: [GetConnectedItemCount( , , Employs, Person)]
```

Get some connection information.

```
Execute: [AssignConnection( , , Employs, Person, "Dellany")]
```

Assign a connection.

```
Execute: [ShowItem( , )]
```

Display the item detail of the marked item.

```
Terminate:
```

DDE Examples

Word Definitions

Category

A Commence category name.

Request

A DDE command that returns the specified information.

Execute

A DDE command that executes a specified action.

Windows Instance Identifier

A unique number (also known as a task ID) that identifies a specific copy of an application when several copies are running at the same time. The identifier number is appended to the application name to identify the application. (e.g. Excel4321)

Piping Delimiter

A delimiter used to append multiple DDE parameters in the same command. For most keyboards, this symbol can be found by pressing <SHIFT><\> (where \ is the backslash key).

Delimiter bookmark

If the optional **Delim** parameter is not specified, the returned list is delimited with CR/LF (ASCII codes 13 and 10). The **Delim** parameter can be any string (up to 8 characters).

execute bookmark

Execute Item for GetData and ViewData topics

request ViewData bookmark

Request Item for the ViewData topic

request GetData bookmark

Request Item for the GetData topic

request System bookmark

Request Item for the System topic

See Also

See Also:

Parameter Notes

If the *Category* and *Item* parameters are both blank, then Commence uses the item from the most recent AddItem/AddSharedItem command, MarkActiveItem or ViewMarkItem REQUEST.

ClarifyItem

Whenever possible, use a clarified item name.

V100

Version 1.00

V100a

Version 1.00a

V200

Version 2.00

V210

Version 2.10

V300

Version 3.00