

TaskAide Scripting Reference

The scripting engine used in TaskAide is VBScript. You can use intrinsic objects in VBscript or create references to ActiveX objects such as the ADO data connection objects, File System Object etc.

For an in-depth reference on VBScript see the Microsoft web site.

TaskAide passes several objects in to the script that can be used from within the script.

TASK

The task object contains properties relating to the current task where the script is executing. In scripted tasks it's used to set the next transition for the system to take. In user assignment scripts it's used to set the current user account. (IE assign the task to the user.)

It has the following properties:

TaskName

Returns the name of the current task.

TaskID

Returns the system ID number for the current task.

FlowName

Returns the name of the flow definition that the current task is a member of.

FlowDefinitionID

Returns the system ID number for the flow definition.

ImagingFolderName

Returns the name of the imaging folder this flow is defined for.

FlowInstanceID

Returns the system ID number for the flow instance.

CurrentUser

Used to assign a task to a user. This is only used in a "User Assignment" script. For a scripted task it contains the value "SYSTEM".

Example:

```
Task.CurrentUser = "joe"
```

This will assign the current task to user account "joe".

SelectedTransitionID

Used to tell the system which transition to take to the next task. Set this to the transition ID number as listed in the task properties.

Example:

```
Task.SelectedTransitionID = 35
```

This will cause the system to select transition ID 35 when the script finishes executing. If this value is left at 0, then the scripting engine will not take any transition and will leave the flow at the current task. This will cause the script to execute again next time the transition daemon is run. Setting the SelectedTransitionID=0 can be useful in cases where you need the flow to wait at this task until a certain condition is met.

LastWorkedBy

This property returns a Taskaide user object that represents the user that worked the last manual task in this flow. This can be used after an approval/denial task to get which actual user performed the approval or denial. You can then update a database field or do other work based on the user account.

Example:

```
ImagingRecord("ApprovedBy") = Task.LastWorkedBy.UserName
```

TASKERROR

This object is used to flag an error. Setting the error property on this object will cause the flow to enter an error state.

The object can also be used for debugging scripts by using it's logging methods.

It has the following properties and methods:

ErrorFlag

Set this to True to set the flow to an error state.

ErrorDescription

Use this to set a descriptive error message.

LogFile

Use this to specify a log file to write log entries to. By default it is set to [TACore.DLL path]\TAScript.log. (IE: C:\Program Files\ESCCCommonFiles\TAScript.log)

You can change this to have different tasks write to different log files for debugging purposes.

WriteLog(LogEntry)

Use this method to output an entry to the log file. This can be used to log progress through a script or to output values from a script for debugging purposes.

Example:

```
TaskError.WriteLog "Checking Value X = " & X
```

EMAIL

The e-mail object can be used to send basic e-mails from a script. This object adds e-mail messages to the Task Aide Mail queue.

Email_Add(Recipient, Subject, MessageBody)

Use this method to send an e-mail.

For the Recipient parameter, you can enter the following values:

“Admins” – sends the message to any user marked as an admin in TaskAide

“Role:[RoleName]” – Sends the message to all members of a role.

“User:[UserName]” – Sends the message to a specific user account.

“[UserAccount]” – Sends the message to a specific user account.

“Field:[Field Name]” – Sends the message to the e-mail address or user name contained in the specified imaging field.

“[specific e-mail address]” – Sends the message to specific email address such as

“someone@domain.com”

The “Subject” parameter is a test value for the message subject.

The “MessageBody” parameter is the text of the message body. HTML tags can be used here to format the message.

Mail message are submitted to the mail queue after the script completes.

Examples:

```
Email.Email_Add "joe@company.com", "Here is a test message", "Please read this test message."
```

```
Email.Email_Add "ROLE:General Manager", "This is an e-mail notification", "Please reveiew the following.<BR>Item: 1..."
```

```
Email.Email_Add "User:jsmith", "Test Message", "Here is a test message. <B>Text in bold</B>"
```

```
Email.Email_Add "Field:AddedBy", "Test Message", "The request you added has been completed."
```

Email_Clear()

Use this method to clear any messages that were previously added.

IMAGINGRECORD

The ImagingRecord object contains the field values from the imaging system for the record associated with the current flow.

This object is a Microsoft ADO RecordSet object containing a single record. This object is updateable meaning you can set field values and the imaging database will be updated after the script completes or when you call the Update method of the record set. For more information on Microsoft ADO RecordSet objects see the documentation at www.microsoft.com

To get a value from the record set use parenthesis with the field name in quotes.

Example: Getting a value from the imaging record.

```
InvoiceAmt = ImagingRecord("INV_AMT")
```

Example: Sending an e-mail using a value from the imaging record.

```
Email_Add "joe@company.com", "Invoice Number " & _  
ImagingRecord("INV_Number") & " has been approved.", "Message text"
```

Example: Setting a value in the imaging record.

```
ImagingRecord("ApprovedBY") = "Bill"
```

You can call the update method here to immediately write the data to the database.

```
ImagingRecord.Update
```

Note: The system automatically calls the update method after the script completes.

DB

The DB object contains references to Microsoft ADO Connection objects. It has properties representing the connection to the TaskAide database, the imaging master database, and the imaging folder database.

DBTaskAide

Represents the connection to the TaskAide Database. ([workflow prefix]Master)

DBImaging

Represents the connection to the Imaging Database. (IMEV_Master)

DBImgingFolder

Represents the connection to the Imaging Folder Database. (IMEV_[FolderName])

These objects can be used to perform queries against databases. These are passed in to the script primarily as a convenience to save you the trouble of having to create the ADODB.Connection object yourself.

Taskaide Core Objects

Taskaide Core Objects can be used from within your scripts. The objects are not passed in to the scripting engine by default. What this means is that in order to use them you will need to create an instance of the object before using it. To create an instance of the object, dim the variable, then use the CreateObject() function.

Example:

```
Dim taUser  
  
Set taUser = CreateObject("TACore.User")
```

The Taskaide Core Objects can be used for advanced scripting in cases where you need to load user or role data from the taskaide database. All core objects have a LastErrorCode and LastErrorDescription property that can be checked for errors after making a function call.

User

Properties:

`UserID_TaskAide` – Returns the TaskAide user ID for the user.
`UserID_Imaging` – Returns the LaserVault Imaging UserID for the user.
`UserName` – Users imaging account name.
`EmailAddress` – User's e-mail address.
`Roles` – RoleCollection object that contains each role the user is a member of.
`UserProperties` – UserPropertiesCollection object that contains each user property and value defined for this user.

Methods:

```
DB_LoadByUsername(DBConnection As TACore.DatabaseConnection, ByVal  
UserName As String, Optional ByVal IncludeRoles As Boolean, Optional  
ByVal RaiseError As Boolean = True)
```

This method loads the user's information into the object using the user name.

Example:

```
Dim taUser  
Set taUser = CreateObject("TACore.User")  
taUser.DB_LoadByUsername DB.DBTaskAide, "joe", True
```

This would load the properties for user account "joe" and also load the roles the user is a member of.

```
DB_Load(DBConnection As TACore.DatabaseConnection, Optional ByVal  
IncludeRoles As Boolean, Optional ByVal RaiseError As Boolean = True)
```

This method will load a user's account details based on the task aide user id.

Example:

```
taUser.UserID_TaskAide = 1  
taUser.DB_Load DB.DBTaskAide, True
```

UserCollection

Properties:

Count – Number of items in the collection.

Methods:

Item([index, or user name]) – Default method that returns the selected item

Example:

```
Set User = UserCollection.Item("joe")  
Set User = UserCollection("joe")
```

```
Set User = UserCollection.Item(1)  
Set User = UserCollection(1)
```

```
DB_LoadAllTaskAideUsers(DBConnection As TACore.DatabaseConnection,  
Optional ByVal IncludeRoles As Boolean, Optional ByVal RaiseError As  
Boolean = True)
```

This method loads all users defined in TaskAide. For the DBConnection parameter you can use the DB object passed in to the script.

Example:

```
Dim taUsers
```

```
Set taUsers = CreateObject("TACORE.UserCollection")
```

```
taUsers.DB_LoadAllTaskAideUsers DB, True
```

```
DB_LoadAdmins(DBConnection As TACore.DatabaseConnection)
```

This method loads all users marked as an admin in the TaskAide system.

Example:

```
Dim taUsers  
Set taUsers = CreateObject("TACore.UserCollection")  
taUsers.DB_LoadAdmins DB
```

UserProperty

This object represents on user property associated with a user.

Properties:

PropertyName – Name of the user property.
PropertyValue – Value of the user property.

UserPropertyCollection

This is a collection of user property objects.

Properties:

Count – Number of items in the collection

Methods:

Item([Index or property name]) – Returns an item in the collection.

Example:

```
Dim taUserProp  
  
Set taUserProp = UserProperties.Item("Department")  
  
Exists(PropertyName as String) as Boolean
```

Returns true if a property with the given name exists in the collection.

Role

The role object represents a defined role in the system.

Properties:

`RoleName` – Name of the role.

`RoleDescription` – Description of the Role

`Users` – `UserCollection` containing all the users in the role.

RoleCollection:

This object represents a list of Role objects.

Properties:

`Count` – Number of items in the collection.

Methods:

`Item([Index or Role Name])` – Returns the specified role from the collection.

Example:

```
Dim taRole
```

```
Set taRole = Roles.Item("General Manager")
```

```
Set taRole = Roles.Item(1)
```

```
DB_LoadAllRoles(DBConnection As TACore.DatabaseConnection, Optional  
ByVal IncludeUsers As Boolean, Optional ByVal RaiseError As Boolean =  
True)
```

This function will load all roles defined in the system. You can pass the DB object for the `DBConnection` parameter.

Example:

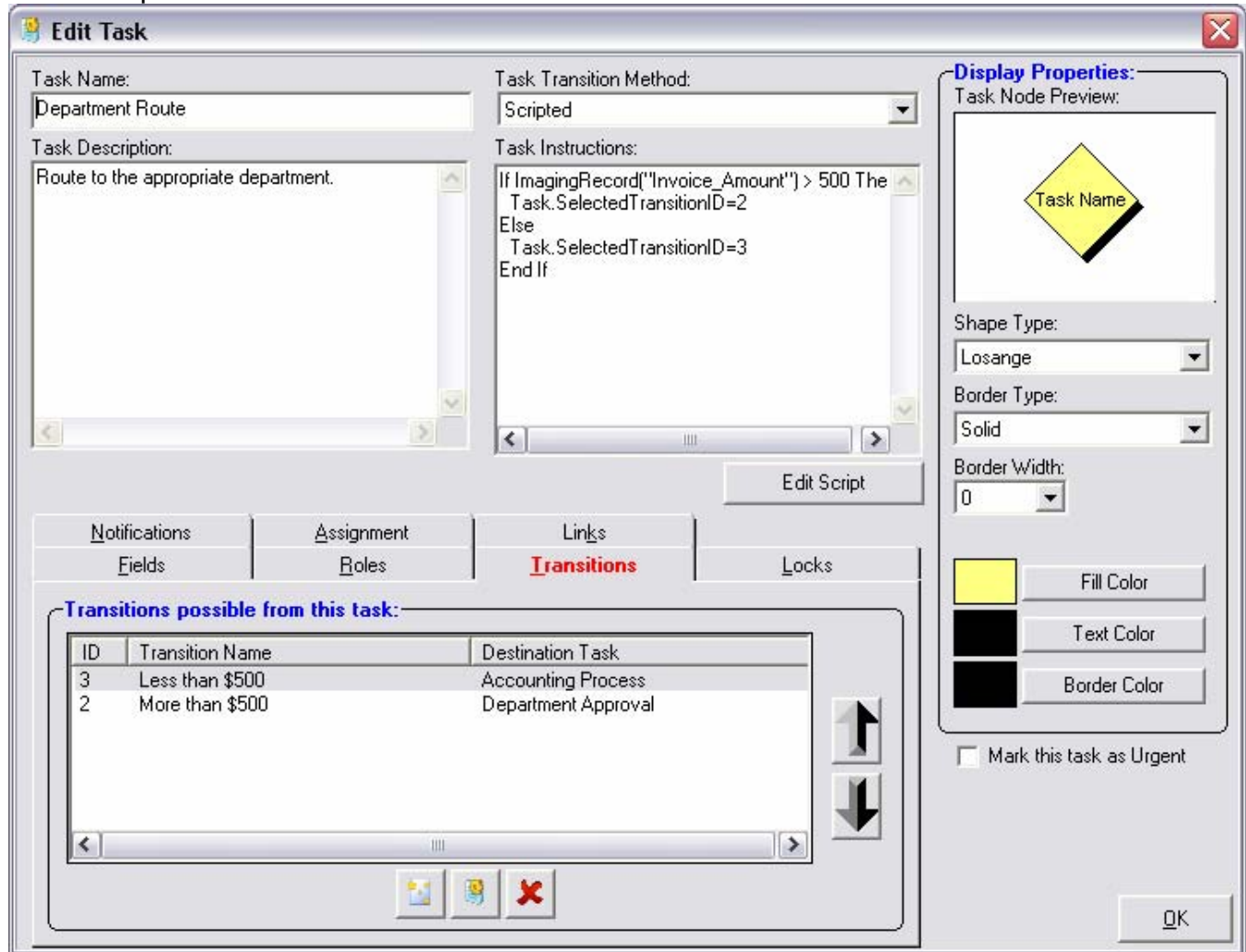
```
Dim taRoles
```

```
Set taRoles = CreateObject("TACore.RoleCollection")
```

```
taRoles.DB_LoadAllRoles DB, True
```

Scripting Examples

The example below shows how to route an invoice based on the invoice amount.



We've created a task called "Department Route" and selected the "Scripted" transition method.

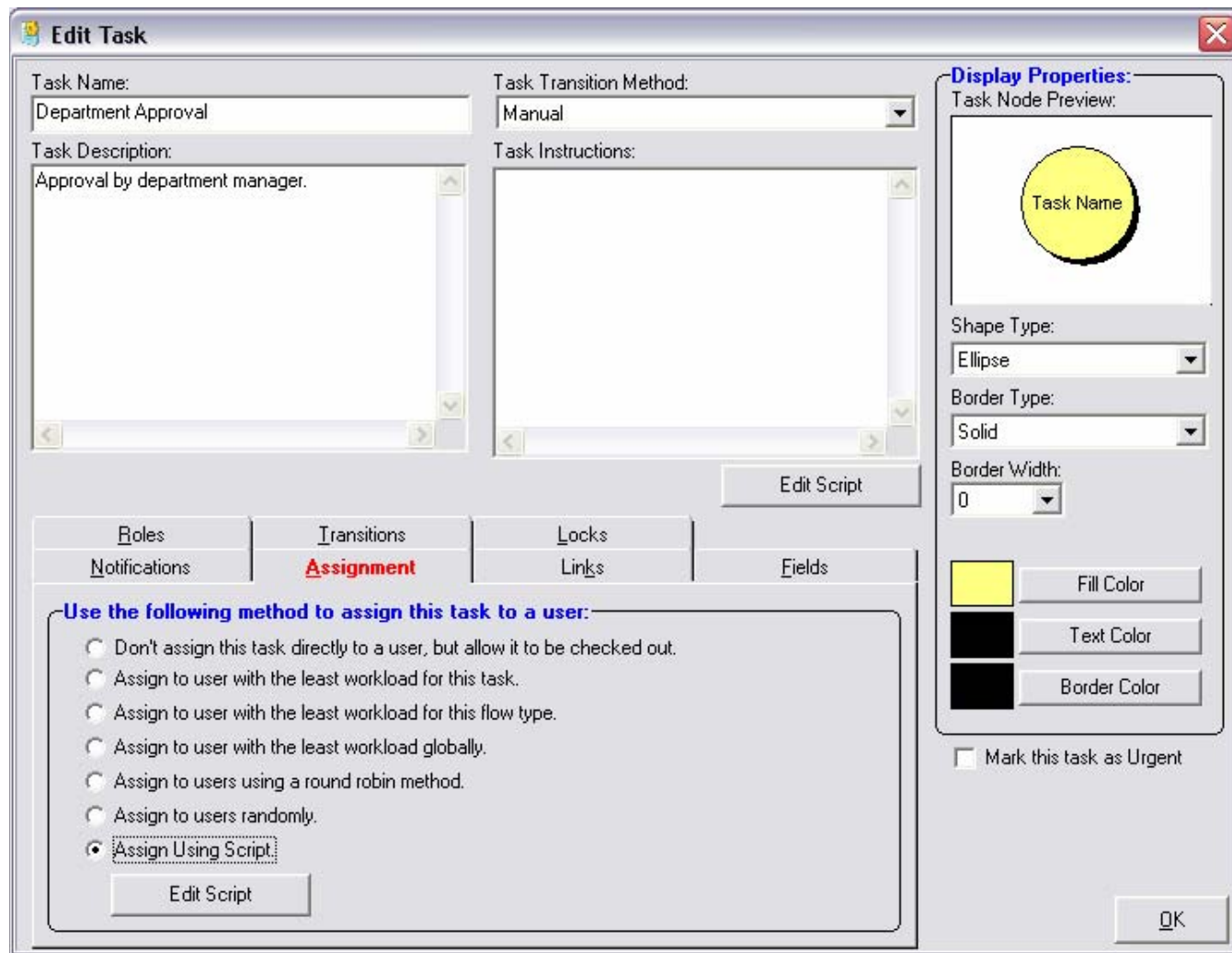
For the instructions we've entered the following VBScript. Note: you can click the "Edit Script" button to bring up a full screen window for editing the script.

```
If ImagingRecord("Invoice_Amount") > 500 Then
  Task.SelectedTransitionID=2
Else
  Task.SelectedTransitionID=3
End If
```

In this example we are using the "Invoice_Amount" field from the imaging system to determine where to go next. You can see in the screen shot we have to 2 transitions defined.

The ID number is what's used by the system to determine which transition we take when we set TASK.SelectedTransitionID=[transition id]

The next example shows how to assign a task to a user using script.



In the task properties we have set the user assignment method to “Assign Using Script”. Click the “Edit Script” button on the Assignment pane to edit the user assignment script.

In our example we have the following script defined:

```
If ImagingRecord("State") = "TX" Then
    Task.CurrentUser = "Joe"
Else
    Task.CurrentUser = "Pam"
End IF
```

In this example we are assigning the task based on the “State” field from the imaging system. If the state is TX – Texas, the task is assigned to Joe, else it’s assigned to Pam.

The next scripting example shows how to assign a task to a user based on the department field using a select case statement. We use the "UCase" function to convert the value to upper case so we can type our literal values in upper case to ensure a match.

```
Select Case UCase(ImagingRecord("Department"))  
  
Case "IT"  
    Task.CurrentUser = "Bill"  
  
Case "SALES"  
    Task.CurrentUser = "Joe"  
  
Case "HR"  
    Task.CurrentUser = "Mary"  
  
Case Else  
    Task.CurrentUser = "John"  
  
End Select
```

The next example shows how to update an imaging record with the user name of the person who performed the last task. In this example we have a simple Approve/Deny task for a purchase request. After the request has been approved, we want to fill in our "ApprovedBy" imaging field with the user name of the person who approved the request.

```
`Set the ApprovedBy Field with the user name of the last user to work  
this task.  
ImagingRecord("ApprovedBy") = Task.LastWorkedBy.UserName  
  
`Move to the next task  
Task.SelectedTransitionID = 25
```

The next example shows how to select a role based on the department field and then assign the task to the first user in that role.

```
Dim taRoles
```

```
Dim taRole
```

```
'Create the role collection
```

```
Set taRoles = CreateObject("TACore.RoleCollection")
```

```
'Load all defined roles
```

```
taRoles.DB_LoadAllRoles DB, True
```

```
'Based on the Department field, select a role
```

```
Select Case UCase(ImagingRecord("Department"))
```

```
Case "IT"
```

```
    Set taRole = taRoles("IT Managers")
```

```
Case "HR"
```

```
    Set taRole = taRoles("HR Managers")
```

```
Case "SALES"
```

```
    Set taRole = taRoles("Sales Managers")
```

```
Case Else
```

```
    Set taRole = taRoles("General Managers")
```

```
End Select
```

```
'Assign the task to the first user in that role.
```

```
Task.CurrentUser = taRole.Users(1).Name
```