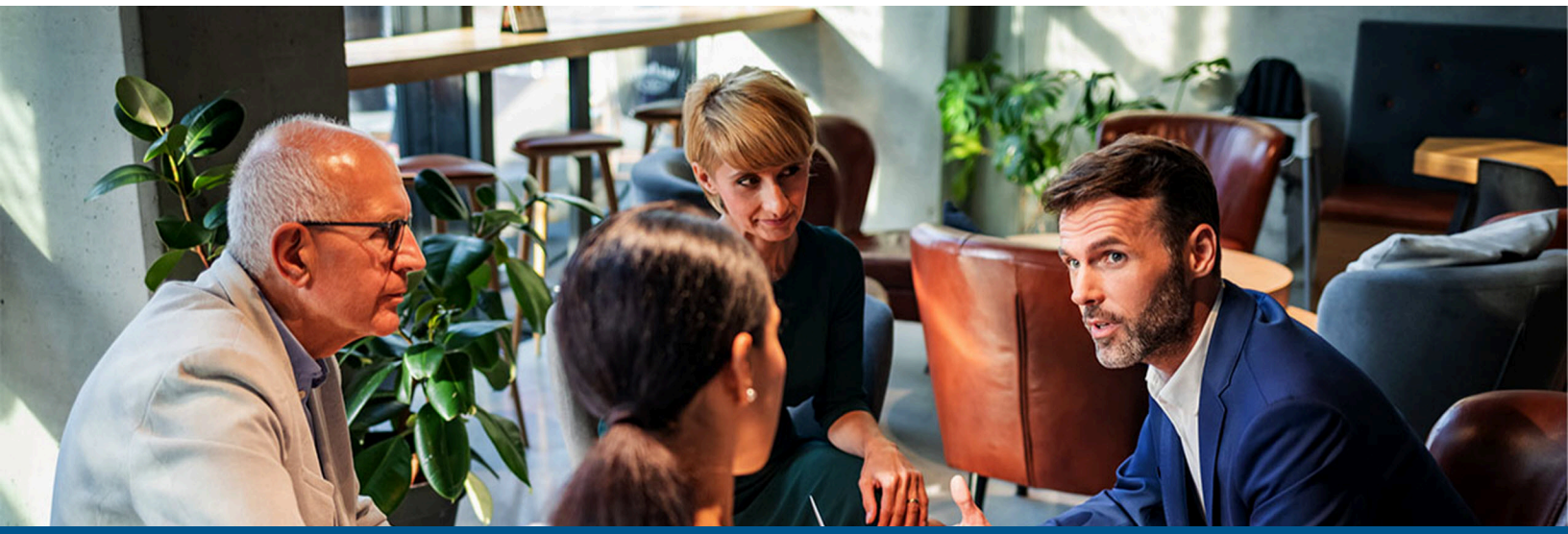


Kofax MarkView
Administrator's Guide, Volume 2

Version: 10.5.0

Date: 2023-12-01

KOFAX

Table of Contents

Preface

Use this document to maintain MarkView components that are administered outside of the MarkView interface. This guide covers advanced administration tasks including descriptions of MarkView Custom Packages and Join Points.

Who should read this guide

Only system administrators with advanced knowledge should use this guide. Administrators must have completed a minimum of Level I training. Some tasks in this guide required a higher level of training. Kofax offers Level II and Level III training to meet those requirements. For information about training, contact Kofax Education Services.

See the *MarkView Administrator's Guide, Volume 1* for a description of administrative roles and the required skill sets associated with those roles.

Related documentation

The documentation set for Kofax MarkView is available online:[1]

https://docshield.kofax.com/Portal/Products/MarkView/10.5.0-yw2qf8m7r6/MarkView.htm

In addition to this guide, the documentation set includes the following items:

Kofax MarkView Features Guide

Use this guide to learn about the features included and options available with MarkView; to become familiar with MarkView products; and to decide which are important to the business challenges you face and best suit your site. This guide includes information about how features impact the workflow, the interaction between features, the touch points with the ERP system, and how features address business problems.

[1] You must be connected to the Internet to access the full documentation set online. If the security policy for your organization requires offline access (without an Internet connection), see the Installation Guide.

Kofax MarkView Planning Guide

Use this guide to learn about the prerequisites for implementing MarkView products. This guide includes system information, such as the protocols required for communication between servers, hardware and software prerequisites, and minimum RAM requirements.

Use this guide in conjunction with the *Kofax MarkView Technical Specifications* document on the Kofax MarkView Product Documentation site to prepare a site for product installation.

Kofax MarkView Installation Worksheet

Use this worksheet to collect and record the information you need to install or upgrade MarkView products.

Kofax MarkView Installation Guide

Use this guide in conjunction with the *Kofax MarkView Installation Worksheet* to install and configure MarkView products and to configure third-party products that integrate with MarkView.

Kofax MarkView Upgrade Guide

Use this guide in conjunction with the *Kofax MarkView Installation Worksheet* to upgrade and configure MarkView products.

Kofax MarkView Integration Guide

Use this guide in conjunction with the *Kofax MarkView Technical Specifications* document on the Kofax MarkView Product Documentation site to learn about the prerequisites for implementing Kofax products and preparing a site for product installation.

Kofax MarkView Reintegration Guide for Upgrades to Oracle E-Business Suite R12 or 12.2

Use this guide to reintegrate MarkView after an upgrade to Oracle E-Business Suite R12 or 12.2.

Kofax MarkView Administrator's Guide, Volume 1

Use this guide to administer the MarkView system. This guide describes how to configure and maintain the applications, solutions, and users that make up the MarkView Suite. The guide also describes how MarkView influences the administration of other servers and software that interface with MarkView applications.

The MarkView Administrator should be well-versed in database administration, application server setup, tuning and maintenance, or should know where to get such information. The administrator's guide does not replicate this information, but conveys MarkView product-specific information.

Kofax MarkView Release Notes

Use this document to learn what is new with the latest MarkView release, identify outstanding defects and workaround solutions where applicable, and learn which defects the release fixes.

Kofax MarkView Technical Specifications
Use this document to learn about supported operating systems and other system requirements.

Getting help with Kofax products

The Kofax Knowledge Portal repository contains articles that are updated on a regular basis to keep you informed about Kofax products. We encourage you to use the Knowledge Portal to obtain answers to your product questions.

To access the Kofax Knowledge Portal, go to https://knowledge.kofax.com.

The Kofax Knowledge Portal is optimized for use with Google Chrome, Mozilla Firefox, or Microsoft Edge.

The Kofax Knowledge Portal provides:

- Powerful search capabilities to help you quickly locate the information you need.
  Type your search terms or phrase into the **Search** box, and then click the search icon.
- Product information, configuration details and documentation, including release news.
  To locate articles, go to the Knowledge Portal home page and select the applicable Solution Family for your product, or click the View All Products button.

From the Knowledge Portal home page, you can:

- Access the Kofax Community (for all customers).
  On the Resources menu, click the **Community** link.
- Access the Kofax Customer Portal (for eligible customers).
  Go to the Support Portal Information page and click **Log in to the Customer Portal**.
- Access the Kofax Partner Portal (for eligible partners).
  Go to the Support Portal Information page and click **Log in to the Partner Portal**.
- Access Kofax support commitments, lifecycle policies, electronic fulfillment details, and self-service tools.
  Go to the Support Details page and select the appropriate article.

Chapter 1

MarkView custom packages

Custom packages are areas of the product to which you can add non-standard code, for example:

- *_CUSTOM PL/SQL packages
- new non-product rule code
- new non-product virtual property code

Custom packages are empty except for minimal coding to make them functional.

Modify retained custom package bodies

Custom package bodies must have the same definition as the shipped product specification. Once you write your implementation of the package body, compile it in the MarkView schema.

Custom packages

Upgrading MarkView overwrites all _CUSTOM packages. See the *MarkView Upgrade Guide* for information about restoring custom packages after upgrades.

AP_WEB_OA_CUSTOM

A custom package provided by Oracle and modified for integrating the MarkView Expense solution.

Functionality

Calling Application, procedure called, variable names

MV_DOCUMENT_CUSTOM

Use this package to support EDI on Demand rendering only.

This package is called every time a document is opened. If you use this package to dynamically render the document the first time it is used, avoid rendering the document again when it is opened on subsequent occasions. If multiple users might open the document simultaneously, provide a locking mechanism to avoid simultaneous conflicting changes.

Do not use this custom package to modify document-level data, such as a document description or document type.

Do not use this custom package to restrict document access by raising an error. Implement custom document access security using procedures and functions in the MV_SECURITY_CUSTOM package body.

Do not explicitly commit or roll back changes in this procedure. Return status as follows to commit or roll back changes:

- **NO_CHANGES**: No changes made: no commit or roll back is performed (default).

  The user will see any changes that were made when the user opens document. If the user performs an action in the viewer that causes a commit (such as locking a view or taking an action), the changes might be committed. However, whether or not the commit occurs depends on what the user does once the document is open. To ensure predictable behavior, return a status of COMMIT_CHANGES if any changes are in this function.
- **ROLLBACK_CHANGES**: Roll back changes made in the PreOpenDocument function. The roll back is to a savepoint immediately before the function call. Changes made outside this function are not rolled back.
- **COMMIT_CHANGES**: Commit changes made by this function. Must be used whenever changes such as adding pages, rendering, adding markups programmatically are made by code in this procedure.

The **DocumentData_T** record includes information about the current document. The following record structures are passed in to the procedure.

Parameters for DocumentData_T Record Structures	
Parameter	**Type**
DocumentID	mv_document.document_id%TYPE
DocumentTypeID	mv_document.document_type_id%TYPE
DocumentTypeName	mv_document_type.document_type_name%TYPE
Description	mv_document.description%TYPE
BatchNumber	mv_document.batch_number%TYPE
CreationUserID	mv_document.creation_user_id%TYPE
CreationTimestamp	Date

The **EnvironmentData_T** record includes information about the environment in which the document is accessed. The following record structures are passed in to the procedure.

Parameters for EnvironmentData_T Record Structures	
Parameter	**Type**
WorkstationSerialNumber	mv_workstation.serial_number%TYPE
PlatformID	mv_platform.platform_id%TYPE
PlatformName	mv_platform.platform_name%TYPE
NO_CHANGES	constant number := 0
ROLLBACK_CHANGES	constant number := 1
COMMIT_CHANGES	constant number := 2

Function PreOpenDocument

The PreOpenDocument function is called when a document is opened after locking the document and checking that access to the document is authorized, but before returning any information to the viewer about the pages and markups.

Use the function to modify the document structure and contents. For example, pages may be added using procedures and functions in the MV_DOC package and markups can be placed or moved using procedures and functions in the MV_MARKUP and MV_RENDER packages.

Parameters for the PreOpenDocument Function		
Parameter	**Input/Output**	**Type**
UserID	IN	Varchar2
DocumentData	IN	DocumentData_T
EnvironmentData	IN	EnvironmentData_T
return		number

MV_FOLDER_CUSTOM

- **ValidateDocumentIndex**: Called once for each document as it is indexed when the user chooses document indexing.

Parameters for the ValidateDocumentIndex Procedure		
Parameter	**Input/Output**	**Type**
DocumentTypeID	IN	Number
DocumentDescription	IN	Varchar2
IsValid	OUT	Boolean
ErrorMessage	OUT	Varchar2

  DocumentTypeID represents the DocumentType that the user has chosen. The DocumentDescription represents the value that the user has entered for indexing (if any).

  - Set IsValid to FALSE if the DocumentDescription is not valid for the given DocumentType.
  - Set ErrorMessage to the string to be shown to the user to indicate that the index is not valid.
  - Set IsValid to TRUE if the DocumentDescription variable is valid for the given DocumentType (ErrorMessage is ignored in this case).
- **PostCreateDocument**: Called once after each new folder document is saved to the database and before it is committed. If Description is changed, set UpdateDocument to True. ImageIDList, GenerateFilenameList, and FileSizeList list the Image ID, Generate Filename, and FileSize of each page in the new document starting with Page Number 1 and continuing for each page in page order.

Table	Index
type NumberList_T is table of Number	index by binary_integer

Table	Index
type CompleteFilenameList_T is table of MV_Scheme.CompleteFilename_T	index by binary_integer

If Description changes, set UpdateDocument to True. ImageIDList, GenerateFilenameList, and FileSizeList list the Image ID, Generate Filename, and FileSize of each page in the new document starting with Page Number 1 and continuing for each page in page order.

Parameters for the PostCreateDocument Procedure		
Parameter	**Input/Output**	**Type**
FolderID	IN	Number
DocumentID	IN	Number
CreationUserID	IN	Varchar2
DocumentTypeID	IN	Number
Description	IN OUT	Varchar2
SourceFilename	IN	Varchar2
PageCount	IN	Number
ImageIDList	IN	NumberList_T
GeneratedFilenameList	IN	CompleteFilenameList
FileSizeList	IN	NumberList_T
UpdateDocument	OUT	Boolean

- **PostRenameDocument**: Called once after a folder document is renamed and before it is committed.
  - To allow renaming, set AllowRename to True; to deny renaming, set AllowRename to False.
  - If you set AllowRename to False, set ErrorMessage to an appropriate error message.
  - To override the new document description, set NewDocumentDescription to the new description and set UpdateDocument to True.

Parameters for the PostRenameDocument Procedure		
Parameter	**Input/Output**	**Type**
FolderID	IN	Number
DocumentID	IN	Number
RenameUserID	IN	Varchar2
DocumentTypeID	IN	Number
OldDocumentDescription	IN	Varchar2
NewDocumentDescription	IN OUT	Varchar2
UpdateDocument	OUT	Boolean
AllowRename	OUT	Boolean
ErrorMessage	OUT	Varchar2

- **PostRecategorizeDocument**: Called once after a folder document is changed and before it is committed. OldDocumentTypeID represents the document ID before the user made changes. NewDocumentTypeID represents the document id that the user specified for the recategorization.
  - If you change DocumentDescription, set UpdateDocument to True.
  - To allow the change set AllowChange to True.
  - To prevent the change set AllowChange to False. If you set AllowChange to False, set ErrorMessage to an appropriate error message.

Parameters for the PostRecategorizeDocument Procedure		
Parameter	**Input/Output**	**Type**
FolderID	IN	Number
DocumentIn	IN	Number
DocumentDescription	IN OUT	Varchar2
RecategorizeUserID	IN	Varchar2
OldDocumentTypeID	IN	Number
NewDocumentTypeID	IN	Number
Description	IN	Varchar2
UpdateDocument	OUT	Boolean
AllowRecategorize	OUT	Boolean
ErrorMessage	OUT	Varchar2

- **PostUnfolderDocument**: Called once after a folder document is unfoldered and before it is committed.

  To allow the change, set AllowUnfolder to True. To prevent the change, set AllowUnfolder to False. If you set AllowUnfolder to False, set ErrorMessage to an appropriate error message.

Parameters for the PostUnfolderDocument Procedure		
Parameter	**Input/Output**	**Type**
FolderID	IN	Number
DocumentID	IN	Number
DocumentDescription	IN	Varchar2
UnfolderUserID	IN	Varchar2
DocumentTypeID	IN	Number
AllowUnfolder	OUT	Boolean
ErrorMessage	OUT	Varchar2

- **OnSelectDocDescription**: Called once for each document in a folder when the folder is shown. You can use this function to substitute values for the default document values.

Parameters for the OnSelectDocDescription Function		
Parameter	**Input/Output**	**Type**
FolderID	IN	Number
DocumentID	IN	Number
DocumentDescription	IN	Varchar2
DocumentTypeID	IN	Number
DocumentTypeDescription	IN	Varchar2
DocumentDate	IN	Date
return		Varchar2

- **OnSelectDocTypeDescription**: Called once for each document in a folder when the folder is shown. Use this function to substitute values for the default document values.

  This function has the following purity restrictions:

  PRAGMA RESTRICT_REFERENCES (OnSelectDocDescription, WNDS, WNPS

Parameters for the OnSelectDocTypeDescription Function		
Parameter	**Input/Output**	**Type**
FolderID	IN	Number
DocumentID	IN	Number
DocumentDescription	IN	Varchar2
DocumentTypeID	IN	Number
DocumentTypeDescription	IN	Varchar2
DocumentDate	IN	Date
return		Varchar2

- **OnSelectDocDate**: Called once for each document in a folder when the folder is shown. Use this function to substitute values for the default document values.

  This function has the following purity restrictions:

  PRAGMA RESTRICT_REFERENCES (OnSelectDocDate, WNDS, WNPS);

Parameters for the OnSelectDocDate Function		
Parameter	**Input/Output**	**Type**
FolderD	IN	Number
DocumentID	IN	Number
DocumentDescription	IN	Varchar2
DocumentTypeID	IN	Number
DocumentTypeDescription	IN	Varchar2
DocumentDate	IN	Date
return		

MV_SECURITY_CUSTOM

Deprecated for DTM (mvw_servlet_dtm_util), viewer (mv_viewer), and Web Utility (mvt_web_utility, mvt_web_utility_ul)

type DocumentData_T is record

Parameter	Type
DocumentID	mv_document.document_id%TYPE
DocumentTypeID	mv_document.document_type_id%TYPE
DocumentTypeName	mv_document_type.document_type_name%TYPE
Description	mv_document.description%TYPE
BatchNumber	mv_document.batch_number%TYPE
CreationUserID	mv_document.creation_user_id%TYPE
CreationTimestamp	Date

type EnvironmentData_T is record

Parameter	Type
WorkstationSerialNumber	mv_workstation.serial_number%TYPE
PlatformID	mv_platform.platform_id%TYPE
PlatformName	mv_platform.platform_name%TYPE

Valid return values for AuthorizeUserDocumentAccess:

Do not change these values.

Parameter	Type	Value
MV_SECURITY_UTIL.ACCESS_GRANTED	CONSTANT	Number: = -1;
MV_SECURITY_UTIL.ACCESS_DENIED	CONSTANT	Number: = NULL;
MV_SECURITY_UTIL.ACCESS_DEFAULT	CONSTANT	Number: = 0;

- **AuthorizeUserDocumentAccess**: Implements custom security policies governing document access.

Parameters for the AuthorizeUserDocumentAccess Function		
Parameter	**Input/Output**	**Type**
UserID	IN	Varchar
DocumentData	IN	DocumentData_T
EnvironmentData	IN	EnvironmentData_T
RunProductCode	OUT	Boolean

Parameters for the AuthorizeUserDocumentAccess Function		
Parameter	**Input/Output**	**Type**
return		Number

This function must return one of the following values:

- ACCESS_GRANTED: Grants the user access regardless of User Groups ACCESS privileges.
- ACCESS_DENIED: Denies the user access regardless of User Groups ACCESS privileges.
- ACCESS_DEFAULT: Grants or denies the user access based on the User Group ACCESS privileges. If the user belongs to a user group that enables the ACCESS privilege for the relevant Document Type, the user can access the document.

This function has the following purity restrictions:

PRAGMA RESTRICT_REFERENCES (AuthorizeUserDocumentAccess, WNDS, WNPS);

The code in this function cannot write to the database ("Writes No Database State") or the package state (package variables "Writes No Package State") and attempts fail with a "subprogram violates is associated pragma" error.

This function also cannot use raise_application_error due to the purity constraints on the package.

- **AuthorizeUserDocumentRename**: Implements custom security policies for the document rename privilege.

Parameters for the AuthorizeUserDocumentRename Function		
Parameter	**Input/Output**	**Type**
UserID	IN	Varchar2
DocumentData	IN	DocumentData_T
EnvironmentData	IN	EnvironmentData_T
RunProductCode	OUT	Boolean
return		Number

This function must return one of the following:

- RENAME_GRANTED: Allows the user to rename documents regardless of User Group RENAME privileges.
- RENAME_DENIED: Does not allow the user to rename documents regardless of User Group RENAME privileges.
- RENAME_DEFAULT: Grants or denies the user renaming rights based on User Group RENAME privileges. If the user belongs to a user group that enables the RENAME privilege for the relevant Document Type, the user can rename the document.

This function has the following purity restrictions:

PRAGMA RESTRICT_REFERENCES (AuthorizeUserDocumentRename, WNDS, WNPS);

The code in this function cannot write to the database ("Writes No Database State") and cannot write to package state (package variables "Writes No Package State"). Attempts to do so fail at compile time with "subprogram violates is associated pragma" error.

This function cannot use raise_application_error due to the purity constraints on the package.

PRAGMA RESTRICT_REFERENCES (AuthorizeUserDocumentRename, WNDS, WNPS);
Valid return values for AuthorizeUserDocumentRename:

Do not change these values.

Parameter	Type	Value
MV_SECURITY_UTIL.RENAME_GRANTED	CONSTANT	Number: = -1;
MV_SECURITY_UTIL.RENAME_DENIED	CONSTANT	Number: = NULL;
MV_SECURITY_UTIL.RENAME_DEFAULT	CONSTANT	Number: = 0;

- **AuthorizeUserDocRecategorize**: Implements custom security policies for the document recategorize privilege.

Parameters for the AuthorizeUserDocRecategorize Function		
Parameter	**Input/Output**	**Type**
UserID	IN	Varchar2
DocumentData	IN	DocumentData_T
EnvironmentData	IN	EnvironmentData_T
RunProductCode	OUT	Boolean
return		Number

Valid return values for AuthorizeUserDocumentRecategorize:

Do not change these values.

Parameter	Type	Value
MV_SECURITY_UTIL.RECATEGORIZE_GRANTED	CONSTANT	Number: = -1;
MV_SECURITY_UTIL.RECATEGORIZE_DENIED	CONSTANT	Number: = NULL;
MV_SECURITY_UTIL.RECATEGORIZE_DEFAULT	CONSTANT	Number: = 0;

This function must return one of the following values:

- RECATEGORIZE_GRANTED: Allows the user to recategorize regardless of User Groups RECATEGORIZE privileges.
- RECATEGORIZE_DENIED: Does not allow the user to recategorize regardless of User Group RECATEGORIZE privileges.
- RECATEGORIZE_DEFAULT: Grants or denies the user the right to recategorize based on the user group RECATEGORIZE privileges. If the user belongs to a user groups that enables the RECATEGORIZE privilege for the relevant Document Type, the user can recategorize the document.

This function has the following purity restriction:
PRAGMA RESTRICT_REFERENCES (AuthorizeUserDocRecategorize, WNDS, WNPS)

The code in this function:

- Cannot write to the database (Writes No Database State).
- Cannot write to package state (package variables Writes No Package State). Attempts to do so fail at compile time with "subprogram violates is associated pragma" error.
- Cannot use raise_application_error.

- **AuthorizeUserDocumentUnfolder**: Implements custom security policies for the document unfolder privileges.

Parameters for the AuthorizeUserDocumentUnfolder Function		
Parameter	**Input/Output**	**Type**
UserID	IN	Varchar2
DocumentData	IN	DocumentData_T
EnvironmentData	IN	EnvironmentData_T
RunProductCode	OUT	Boolean
return		Number

Valid return values for AuthorizeUserDocumentUnfolder

Do not change these values.

Parameter	Type	Value
MV_SECURITY_UTIL.UNFOLDER_GRANTED	CONSTANT	Number: = -1;
MV_SECURITY_UTIL.UNFOLDER_DENIED	CONSTANT	Number: = NULL;
MV_SECURITY_UTIL.UNFOLDER_DEFAULT	CONSTANT	Number: = 0;

This function must return one of following values:

- UNFOLDER_GRANTED: Allows the user to unfolder regardless of User Groups UNFOLDER privileges.
- UNFOLDER_DENIED: Does not allow the user to unfolder regardless of User Groups UNFOLDER privileges.
- UNFOLDER_DEFAULT: Grants or denies the rights to unfolder based on the UNFOLDER privilege grants in the User Group. If the user belongs to a user group that enables the UNFOLDER privilege for the relevant Document Type, the user can unfolder the document.

This function has the following Purity Restriction:

PRAGMA RESTRICT_REFERENCES (AuthorizeUserDocumentUnfolder, WNDS, WNPS);

The code in this function:

- Cannot use raise_application_error due to the purity constraints on the package.
- Cannot write to the database (Writes No Database State).
- Cannot write to package state (package variables Writes No Package State). Attempts to do so fail at compile time with a "subprogram violates is associated pragma" error.

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameter for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVAP_CUSTOM

Creates the specification for the MVAP_Custom package. This package consolidates function calls and code used by the template to support Accounts Payable look-ups. This package also allows specific user customization.

This custom package contains the following procedures and functions.

Procedures	Functions
procedure SupervisorDunning	PackageVersion
procedure NotifyHighPriority	
procedure CreateEmail	
procedure NotifyHighPriority	
procedure CreateEmail	
procedure EmailDocumentLink	
procedure DetermineReviewRequirement	
procedure DetermineQARequirement	
procedure DetermineSFAuditRequirement	
procedure ExternalEntryComplete	
procedure OnPopulate	
procedure GetPackageVersion	

- **SupervisorDunning**: Sends email to the supervisor of the UserID and CCs the User. The message will be defined in the package. Because this is custom, you can modify the message for a given installation. The work item instance ID is passed so that item-specific information can be included in the message. The check for the number of days to wait until sending this dunning message needs to be done outside this procedure.

Parameters for the SupervisorDunning Procedure		
Parameter	**Input/Output**	**Type**
WorkItemInstanceID	IN	Number
UserID	IN	Varchar2

- **NotifyHighPriority**: Sends an email to the email address for the given UserID. This message is currently defined in the package. The work item instance ID is passed so that item-specific information can be included in the message. The procedure must set the RunProductCode output parameter to false to run custom logic. Otherwise the product logic is processed.

Parameters for the NotifyHighPriority Procedure		
Parameter	**Input/Output**	**Type**
WorkItemInstanceID	IN	Number
UserID	IN	Varchar2
RunProductCode	OUT	Boolean

- **CreateEmail**: Creates an email message with the given properties. The procedure must set the RunProductCode output parameter to false to run the custom logic. Otherwise the product logic is processed.

Parameters for the CreateEmail Procedure		
Parameter	**Input/Output**	**Type**
UserID	IN	mv_user_profile.user_id%type
EmailAddress	IN	varchar2
MailSubject	IN	varchar2
MailBody	IN	varchar2
MailFrom	IN	varchar2 (default null)
FromName	IN	varchar2 (default null)
CC	IN	varchar2 (default null)
BCC	IN	varchar2 (default null)
BodyFormat	IN	varchar2 (default null)
AttachBody	IN	varchar2 (default null)
AttachFilename	IN	varchar2 (default null)
RunProductCode	OUT	Boolean

- **CreateEmail2**: Creates an email message for the given user and directs the user to MarkView home. The procedure must set the RunProductCode output parameter to false to run the custom logic. Otherwise, the product logic is processed.

Parameters for the CreateEmail2 Procedure		
Parameter	**Input/ Output**	**Type**
UserID	IN	mv_user_profile.user_id%Type

Parameters for the CreateEmail2 Procedure		
Parameter	**Input/ Output**	**Type**
RunProductCode	OUT	Boolean

- **CreateEmail3**: Creates an email message for the given work item and directs the user to the associated document and to MarkView home. The procedure must set the RunProductCode output parameter to false to run the custom logic. Otherwise, the product logic is processed.

Parameters for the CreateEmail3 Procedure		
Parameter	**Input/ Output**	**Type**
UserID	IN	mv_user_profile.user_id%Type
WorkItemInstanceID	IN	sf_workitem_instance.workitem_instance_id%type
RunProductCode	OUT	Boolean

- **EmailDocumentLink**: Sends an information-only email to the user specified in the Markup. The email contains a link to the document and any details provided in the markup comments. The procedure must set the RunProductCode output parameter to false to run custom logic. Otherwise the product logic is processed.

Parameters for the EmailDocumentLink Procedure		
Parameter	**Input/ Output**	**Type**
WorkItemInstanceID	IN	sf_workitem_instance.workitem_instance_id%type
MarkupObjectID	IN	mv_markup_object.markup_object_id%type
UserID	IN	mv_user_profile.user_id%Type
RunProductCode	OUT	Boolean

- **DetermineReviewRequirement**: Checks the work item against the custom rules to determine if a review is necessary and returns TRUE/FALSE. If TRUE, this procedure must determine if the Review User should be specified and the value of the Review User. The procedure must set the RunProductCode output parameter to false if the ReviewRequired and ReviewUser are to be populated with this custom logic. Otherwise the product logic is processed.

Parameters for the DetermineReviewRequirement Procedure		
Parameter	**Input/ Output**	**Type**
WorkItemItemID	IN	sf_workitem_instance.workitem_instance_id%type
UpdatingUser	IN	sf_user_profile.user_id%type
ReviewRequired	IN OUT	Boolean
ReviewUser	IN OUT	sf_user_profile.user_id%type

- **DetermineQARequirement**: Checks the work item against the custom rules to determine if QA is necessary, and returns TRUE/FALSE. If TRUE, this procedure must determine if the QA User should be specified, and if so, the value of the review user. The procedure must set the RunProductCode

output parameter to false if the QARequired and QAUser are to be populated with this custom logic. Otherwise the product logic is processed.

Parameters for the DetermineQARequirement Procedure		
Parameter	**Input/ Output**	**Type**
WorkItemID	IN	sf_workitem_instance.workitem_instance_id%Type
UpdatingUser	IN	sf_user_profile.user_id%type
QARequired	IN OUT	Boolean
QAUser	IN OUT	sf_user_profile.user_id%type

- **DetermineSFAuditRequirement**: Checks the work item against the custom rules to determine if a Senior Financial Audit is necessary, and returns TRUE/FALSE. If TRUE, this procedure must determine if a Senior Financial Audit User should be specified, and if so, the value of the review user.

Parameters for the DetermineSFAuditRequirement Procedure		
Parameter	**Input/ Output**	**Type**
WorkItemID	IN	sf_workitem_instance.workitem_instance_id%Type
UpdatingUser	IN	sf_user_profile.user_id%type
SFAuditRequired	IN OUT	Boolean
SFAuditUser	OUT	sf_user_profile.user_id%type

- **ExternalEntryComplete**: Takes in the current Work Item Instance ID and the current IsEntryComplete value after the evaluation of the shipped default product code. Allows the SI to define any custom business logic that might need to be implemented to override the default processing logic.

Parameters for the ExternalEntryComplete Procedure		
Parameter	**Input/ Output**	**Type**
WorkItemInstanceID	IN	sf_workitem_instance.workitem_instance_id%type
IsEntryComplete	IN OUT	Boolean

- **OnPopulate**: Returns the LOV data for AP LOVs. The procedure must set the RunProductCode output parameter to false if the LOVData is to be populated with this custom logic. Otherwise the product logic is processed.

Parameters for the OnPopulate Procedure		
Parameter	**Input/Output**	**Type**
LOVData	IN OUT	MV_LOV_Advice.LOVData_T
RunProductCode	OUT	Boolean

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameter for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVAP_INV_RENDER_CUSTOM

- **PreCreateRenderPlaceholder**: Called before the work item associated with an invoice configured to be rendered is created. Allows for the setting of work item properties associated with the new work item.

Parameters for the PreCreateRenderPlaceholder Procedure		
Parameter	**Input/ Output**	**Type**
InvoiceID	IN	varchar2
RenderSource	IN	varchar2
RenderType	IN	varchar2
UserID	IN	varchar2
WorkItemPropertyNameList	OUT	SF_Client.WorkItemPropertyNameList_T
WorkItemPropertyValueList	OUT	SF_Client.WorkItemPropertyNameValueList_T
RunProductCode	OUT	Boolean

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameter for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVAP_NOTIFICATION_CUSTOM

Use this package to customize the subject and body of the email notification for approving and rejecting invoices by email. This custom package lets you customize emails that notify the user about the success or failure of an approval or rejection. This package includes seven PL/SQL functions.

Setting MVAP_ENABLE_NOTIFICATION_EMAIL_CUSTOM to TRUE calls the procedures described in this section. To run with custom logic, these procedures must set the RunProductCode output parameter to FALSE. Otherwise, MarkView logic is run.

Parameters such as MailSubject, MailFrom, CC, BCC and so forth have a size limit of 4000 bytes. The size accepted for the body of an email is unlimited.

- **CreateEmailWithAppAndRej**: Determines the properties of email message, except the body and attachments.

  Parameters for the CreateEmailWithAppAndRej Procedure

Parameter	Input/ Output	Type	Description
WorkItemInstanceID	in	sf_workitem_instance. workitem_instance_id%type	Work item instance id
UserID	in	mv_user_profile.user_id%type	User id, for example, JSMITH
EmailAddress	out	varchar2	Email address for the recipient of the notification
MailSubject	out	varchar2	Subject of the email notification
MailFrom	out	varchar2	Email address for the sender of the notification
FromName	out	varchar2	Name of the sender of the notification
CC	out	varchar2	Email address for recipients who receive a copy of the notification
BCC	out	varchar2	Email address for recipients who receive a blind copy of the notification

Parameter	Input/ Output	Type	Description
RunProductCode	out	boolean	Setting that specifies the logic to run: • FALSE: runs custom logic • TRUE: runs MarkView logic (default)

- **DetermineBodyOfNotification**: Specifies the body of the email message.

  Parameters for the DetermineBodyOfNotification Procedure

Parameter	Input/ Output	Type	Description
WorkItemInstanceID	in	sf_workitem_instance. workitem_instance_id%type	Work item instance id
UserID	in	mv_user_profile.user_id%type	User id, for example, JSMITH
MailBody	out	CLOB	Body of the message
BodyFormat	out	varchar2	Format for the body of the message: • HTML (default) • TEXT
RunProductCode	out	boolean	Setting that specifies the logic to run: • FALSE: runs custom logic • TRUE: runs MarkView logic (default)

- **DetermineAttachOfNotification**: Specifies the format of attachments to the email message, which can be binary or text.

  Parameters for the DetermineAttachOfNotification Procedure

Parameter	Input/ Output	Type	Description
WorkItemInstanceID	in	sf_workitem_instance. workitem_instance_id%type	Work item instance id
UserID	in	mv_user_profile.user_id%type	User id, for example, JSMITH
AttachBody	out	BLOB	Binary data that usually presents an image, but can also be text, music, and so on.

Parameter	Input/ Output	Type	Description
TextAttachBody	out	varchar2	Data that usually presents text. The system can retrieve only one type of text attachment: • AttachBody: a binary attachment • TextAttachBody: a text attachment (limited to 2000 characters)
AttachFileName	out	varchar2	File name of the attachment, for example, picture.pdf or info.txt.
RunProductCode	out	boolean	Setting that specifies the logic to run: • FALSE: runs custom logic • TRUE: runs MarkView logic (default)

- **DoAction**: Performs a specific action on an invoice, such as approval or rejection. Performing of these actions can be customized.

  Parameters for the DoAction Procedure

Parameter	Input/ Output	Type	Description
Subject	in	varchar2	Subject of the email
Body	in	varchar2	Body of the email
Operation	in	MVAP_NOTIFICATION_ATTR. OPERATION%type	Operation to perform: • 1: approval • 0: rejection
RunProductCode	out	boolean	Setting that specifies the logic to run: • FALSE: runs custom logic • TRUE: runs MarkView logic (default)

- **CreateWarningEmail**: Creates a warning email message and sends the message to the MarkView Administrator.

  Parameters for the CreateWarningEmail Procedure

Parameter	Input/ Output	Type	Description
Key	in	varchar2	Security key retrieved from the control email

Parameter	Input/ Output	Type	Description
RightSMTPAdress	in	varchar2	Email address of user who owns the invoice; must also be the sender of the control email
SMTPAdress	in	varchar2	Address retrieved from the control email
Operation	in	MVAP_NOTIFICATION_ATTR. OPERATION%type	Operation performed: • 1: approval • 0: rejection
UserID	in	mv_user_profile.user_id%type	User id, for example, JSMITH
WorkItemInstanceID	in	sf_workitem_instance. workitem_instance_id%type	Work item instance id
RunProductCode	out	boolean	Setting that specifies the logic to run: • FALSE: runs custom logic • TRUE: runs MarkView logic (default)

- **CreateSuccessfulEmail**: Creates an email message notifying the user that the procedure succeeded.

  Parameters for the CreateSuccessfulEmail Procedure

Parameter	Input/ Output	Type	Description
Operation	in	MVAP_NOTIFICATION_ATTR. OPERATION%type	Operation performed: • 1: approval • 0: rejection
UserID	in	mv_user_profile.user_id%type	User id, for example, JSMITH
WorkItemInstanceID	in	sf_workitem_instance. workitem_instance_id%type	Work item instance id
RunProductCode	out	boolean	Setting that specifies the logic to run: • FALSE: runs custom logic • TRUE: runs MarkView logic (default)

- **CreateUnsuccessfulEmail**: Creates an email message notifying the user that the procedure did not succeed.

Parameters for the CreateUnsuccessfulEmail Procedure

Parameter	Input/ Output	Type	Description
Operation	In	MVAP_NOTIFICATION_ATTR. OPERATION%type	Operation performed: • 1: approval • 0: rejection
UserID	in	mv_user_profile.user_id%type	User id, for example, JSMITH
WorkItemInstanceID	in	sf_workitem_instance. workitem_instance_id%type	Work item instance id
ErrorDescription	In	varchar2	Description of the error that occurred while an approval or rejection was being processed.
RunProductCode	out	boolean	Setting that specifies the logic to run: • FALSE: runs custom logic • TRUE: runs MarkView logic (default)

MVERP_CUSTOM

- **GetUserEmailAddress**: Returns the email address for the given user id. The procedure must set the RunProductCode output parameter to false to run the custom logic. Otherwise the product logic is processed.

Parameters for the GetUserEmailAddress Function		
Parameter	**Input/Output**	**Type**
UserID	IN	mv_user_profile.user_id%Type
RunProductCode	OUT	Boolean
return		varchar2

  Activate this package by setting the preference **EMAIL_DOCUMENT_RECIPIENT_LIST_FAST** to FALSE. The default value is TRUE.

- **DetermineQARequirement**: Checks the work item against the custom rules to determine if QA is necessary, and returns TRUE/FALSE based on the results. If TRUE, this procedure must further check to determine if the QA User should be specified, and if so, the value of the review user.

Parameters for the DetermineQARequirement Procedure		
Parameter	**Input/ Output**	**Type**
WorkItemID	IN	sf_workitem_instance.workitem_instance_id%TYPE
UpdatingUser	IN	mv_user_profile.user_id%TYPE
QARequired	OUT	Boolean

Parameters for the DetermineQARequirement Procedure		
Parameter	**Input/ Output**	**Type**
QAUser	OUT	mv_user_profile.user_id%TYPE)

- **DetermineReviewRequirement**: Checks the work item against the custom rules to determine if a review is necessary, and returns TRUE/FALSE based on the results. If TRUE, this procedure must determine if the Review User should be specified, and if so, the value of the review user.

Parameters for the DetermineReviewRequirement Procedure		
Parameter	**Input/ Output**	**Type**
WorkItemID	IN	sf_workitem_instance.workitem_instance_id%TYPE
UpdatingUser	IN	mv_user_profile.user_id%TYPE
ReviewRequired	OUT	Boolean
ReviewUser	OUT	mv_user_profile.user_id%TYPE)

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameter for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVERP_SET_WI_DESCR_CUSTOM

- **SetDescription**: Updates the work item and the attached document descriptions based on the work item class, entity, and the unique key ids passed into the procedure. The procedure must set the RunProductCode output parameter to false to run the custom logic. Otherwise the product logic is processed.

Parameters for the SetDescription Procedure		
Parameter	**Input/ Output**	**Type**
WorkItemID	IN	sf_workitem_instance.workitem_instance_id%type

Parameters for the SetDescription Procedure		
Parameter	**Input/ Output**	**Type**
EntityType	IN	varchar2 default null
UniqueKey1	IN	Varchar2
UniqueKey2	IN	varchar2 default null
UniqueKey3	IN	varchar2 default null
UniqueKey4	IN	varchar2 default null
UniqueKey5	IN	varchar2 default null
UpdatingUser	IN	sf_user_profile.user_id%type
WorkItemClassName	IN	sf_workitem_class.class_name%type
DocumentID	IN	mv_document.document_id%type
RunProductCode	OUT	Boolean

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameter for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVERP_APPROVAL_UTIL_CUSTOM

Previously named "MVT_HIERARCHY_API_CUSTOM"

The following public variables provide error numbers for application errors.

Variable	Constant
E_MAUC_BASE	CONSTANT NUMBER := -20700
E_MAUC_NO_USER_NAME	CONSTANT NUMBER := E_MAUC_BASE - 35
E_MAUC_NO_EMPLOYEE_ID	CONSTANT NUMBER := E_MAUC_BASE - 36

Variable	Constant
E_MAUC_NO_MV_USER_ID	CONSTANT NUMBER := E_MAUC_BASE - 40
E_MAUC_NO_SF_USER_ID	CONSTANT NUMBER := E_MAUC_BASE - 45
E_MAUC_INVALID_PREFERENCE	CONSTANT NUMBER := E_MAUC_BASE - 50

- **VerifyAuthority**: Checks the user's spending limit (authority) against the invoice amount and returns TRUE when the user's spending exceeds the invoice amount and FALSE when spending is less than the invoice amount.

  This procedure is called if the MVERP_HIERARCHY_TYPE is set to CUSTOM. The user's spending limit (Approval Limit) should be derived and returned in the procedure. This value populates the workflow message history.

Parameters for the VerifyAuthority Procedure		
Parameter	**Input/ Output**	**Type**
InvoiceID	IN	varchar2
OrganizationID	IN	varchar2 default null
ApprovalUserID	IN	sf_user_profile.user_id%type
AlternateUsrID	IN	sf_user_profile.user_id%TYPE default null
UserID	IN	sf_user_profile.user_id%TYPE default null
WorkitemInstanceID	IN	sf_workitem_instance.workitem_instance_id%type default null
AppsInstance	IN	varchar2 default null
UserHasAuthority	OUT	Boolean
ApprovalLimit	OUT	varchar2

- **GetNextApprover**: Returns the next approver in a hierarchy. This function is called if the MVERP_HIERARCHY_TYPE is set to CUSTOM.

Parameters for the GetNextApprover Function		
Parameter	**Input/ Output**	**Type**
InvoiceID	IN	varchar2
ApprovalUserID	IN	sf_user_profile.user_id%type
OrganizationID	IN	varchar2 default null
WorkitemInstanceID	IN	sf_workitem_instance.workitem_instance_id%type default null
AppsInstance	IN	varchar2 default null
HierarchyType	IN	varchar2 default null
return		sf_user_profile.user_id%type

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameter for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVERP_DFM_CUSTOM

Previously named "MVOA_DFM_CUSTOM"

Procedures and functions from this package are called if ENABLE_MVERP_DFM_CUSTOM preference is TRUE.

- **DeriveInvoiceID**: Derives the AP Invoice ID associated with the current document.

Parameters for the DeriveInvoiceID Procedure		
Parameter	**Input/Output**	**Type**
DocumentID	IN	number
InvoiceID	OUT	varchar2
OrgID	OUT	varchar2
RunProductCode	OUT	Boolean
IsAuthorized	IN OUT	Boolean
ErrorMessage	IN OUT	varchar2

- **DeriveERPUserData**: Determines the ERP User ID, Application ID and Responsibility ID for the current user.

Parameters for the DeriveERPUserData Procedure		
Parameter	**Input/Output**	**Type**
InvoiceID	IN	varchar2
OrgID	IN	varchar2
MarkViewUserID	IN	mv_user_profile.user_id%type
RunProductCode	OUT	Boolean
ERPUserData	IN OUT	MVERP_Authenticate.ERPUserData_T

Parameters for the DeriveERPUserData Procedure		
Parameter	**Input/Output**	**Type**
ErrorMessage	IN OUT	varchar2

- **GetDefaultAccount**: Sets the default value of the account. This function retrieves the default concatenated account segments whenever the amount is entered on a distribution line and the account is not already entered. This is only relevant when the DFM was created with UseProjects set to False.

Parameters for the GetDefaultAccount Function		
Parameter	**Input/Output**	**Type**
DistributionData	IN	MVERP_APInvoice.Distribution_R
MarkViewUserID	IN	mv_user_profile.user_id%type
RunProductCode	OUT	Boolean
IsValid	IN OUT	Boolean
ErrorMessage	IN OUT	varchar2
return		varchar2

- **GetDefaultAccountForLine**: Sets the default value of the account for the invoice line. This function is called to get the default concatenated account segments whenever the amount is entered on an invoice line and the account is not already entered. This is only relevant when the DFM was created with UseProjects set to False. (Oracle R12 environments only)

Parameters for the GetDefaultAccountForLine Function		
Parameter	**Input/Output**	**Type**
LineData	IN	MVERP_APInvoice.Distribution_R
MarkViewUserID	IN	mv_user_profile.user_id%type
RunProductCode	OUT	Boolean
IsValid	IN OUT	Boolean
ErrorMessage	IN OUT	varchar2
return		varchar2

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body and each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **GetProjectAccountingContext**: Determines the project accounting context for a distribution line. This is only relevant when the DFM was created with UseProjects set to TRUE.

Parameters for the GetProjectAccountingContext Procedure		
Parameter	**Input/Output**	**Type**
DistributionData	IN	MVERP_APInvoice.Distribution_R
InvoiceData	IN	MVERP_APInvoice.Header_R
ERPUserData	IN	MVERP_Authenticate.ERPUserData_T
RunProductCode	OUT	Boolean
ProjectAccounting Context	OUT	Boolean

- **InvoiceToolMapping**: Retrieves the ToolID for the specified Document ID, User ID, Invoice ID, and Org ID.

Parameters for the InvoiceToolMapping Procedure		
Parameter	**Input/Output**	**Type**
DocumentID	IN	number
UserID	IN	varchar2
InvoiceID	IN	varchar2
OrgID	IN	varchar2
ToolID	OUT	number
RunProductCode	OUT	Boolean
IsAuthorized	OUT	Boolean
ErrorMessage	OUT	varchar2

- **ReverseRow**: Determines whether the distribution line should be reversed. This function is called when posted or validated distribution should be deleted. This is only relevant when the DFM was created with EnablePostedLineChanges or AllowPostValidationChanges set to TRUE.

Parameters for the ReverseRow Procedure		
Parameter	**Input/Output**	**Type**
DistributionData	IN	MVERP_APInvoice.Distribution_R
ModifiedRowID	IN	rowid
InvoiceData	IN	MVERP_APInvoice.Header_R
NonRecoverableTaxFlag	IN	varchar2
RunProductCode	OUT	Boolean
ReverseRow	IN OUT	Boolean

- **SetLineTypeDefaults**: Sets the default values for Tax fields when the Line Type is changed to a distribution line. This is only applicable for Oracle environments when the DFM is created with EnableLineTypeDefaults set to TRUE.

Parameters for the SetLineTypeDefaults		
Parameter	**Input/Output**	**Type**
InvoiceData	IN	MVERP_APInvoice.Header_R
DistributionData	IN	MVERP_APInvoice.Distribution_R
ERPUserID	IN	varchar2
ResponsibilityID	IN	number
ApplicationID	IN	number
TaxCode	IN OUT	varchar2
TaxCodeID	IN OUT	number
RunProductCode	OUT	Boolean
IsValid	IN OUT	Boolean
ErrorMessage	IN OUT	varchar2

- **SetLineTypeDefaultsForLine**: Sets the default values for Tax fields when the Line Type is changed to a distribution line. This is only applicable for Oracle environments when the DFM is created with EnableLineTypeDefaults set to TRUE.

Parameters for the SetLineTypeDefaultsForLine		
Parameter	**Input/Output**	**Type**
InvoiceData	IN	MVERP_APInvoice.Header_R
LineData	IN	MVERP_APInvoice.Distribution_R
ERPUserID	IN	varchar2
ResponsibilityID	IN	number
ApplicationID	IN	number
TaxCode	IN OUT	varchar2
TaxCodeID	IN OUT	number
Tax	IN OUT	varchar2
TaxStatus	IN OUT	varchar2
TaxJurisdictionCode	IN OUT	varchar2
RunProductCode	OUT	Boolean
IsValid	IN OUT	Boolean
ErrorMessage	IN OUT	varchar2

- **SetProjectDefaults**: Sets the default values for additional project-related fields when the project distribution line changed. This is only relevant when the DFM is created with UseProjects set to TRUE.

Parameters for the SetProjectDefaults Procedure		
Parameter	**Input/Output**	**Type**
DistributionData	IN	MVERP_APInvoice.Distribution_R

Parameters for the SetProjectDefaults Procedure		
Parameter	**Input/Output**	**Type**
FNDUserID	IN	number
ApplicationID	IN	number
ResponsibilityID	IN	number
Task	OUT	varchar2
TaskID	OUT	number
ExpenditureType	OUT	varchar2
ExpenditureOrganization	OUT	varchar2
ExpenditureOrganizationID	OUT	number
ExpenditureItemDate	OUT	date
PAQuantity	OUT	number
RunProductCode	OUT	Boolean
IsValid	IN OUT	Boolean
ErrorMessage	IN OUT	varchar2

- **SetProjectDefaultsForLine**: Sets the default values for additional project-related fields when the project changed.

Parameters for the SetProjectDefaultsForLine Procedure		
Parameter	**Input/Output**	**Type**
LineData	IN	mverp_apinvoice.Distribution _R
FNDUserID	IN	number
ApplicationID	IN	number
ResponsibilityID	IN	number
Task	OUT	varchar2
TaskID	OUT	number
ExpenditureType	OUT	varchar2
ExpenditureOrganization	OUT	varchar2
ExpenditureOrganizationID	OUT	number
ExpenditureItemDate	OUT	date
PAQuantity	OUT	number
RunProductCode	OUT	Boolean
IsValid	IN OUT	Boolean
ErrorMessage	IN OUT	varchar2

- **ValidateInvoiceLine**: Performs custom validation required for an invoice line. This procedure cannot change values on an invoice line. (Oracle R12 environments only)

Parameters for the ValidateInvoiceLine Procedure		
Parameter	**Input/ Output**	**Type**
UserID	IN	mv_user_profile.user_id%Type
LineData	IN	MVERP_APInvoice.Distribution _R
SegmentValueList	IN	MVERP_DFM_Accounting_Flex.SegmentValueList_T
RunProductCode	OUT	Boolean
IsValid	IN OUT	Boolean
ErrorMessage	IN OUT	varchar2

- **ValidateRow**: Performs custom validation required for a distribution line. This procedure cannot change values on a distribution line. (Oracle R12 environments only)

Parameters for the ValidateRow Procedure		
Parameter	**Input/ Output**	**Type**
UserID	IN	mv_user_profile.user_id%Type
DistributionData	IN	MVERP_APInvoice.Distribution _R
SegmentValueList	IN	MVERP_DFM_Accounting_Flex.SegmentValueList_T
RunProductCode	OUT	Boolean
IsValid	IN OUT	Boolean
ErrorMessage	IN OUT	varchar2

MVERP_DFM_DESCR_FLEX_CUSTOM

Previously named "MVOA_DFM_DESCR_FLEX_CUSTOM"

Procedures and functions from this package are called only if ENABLE_MVERP_DFM_DESCR_FLEX_CUSTOM preference is set to TRUE.

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body and each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PopulateSegment**: Populates the DFF segment for a distribution line. This is only relevant when the DFM is created with DFM_FIELD_DFF field enabled.

Parameters for the PopulateSegment Procedure		
Parameter	**Input/Output**	**Type**
DistributionData	IN	mverp_apinvoice.Distribution_R,
ERPUserID	IN	varchar2
ResponsibilityID	IN	number
ApplicationID	IN	number
DescrFlexSegment	IN	MVERP_DFM_Descr_Flex. DescrFlexSegment_R
DescrFlexContextCode	IN	varchar2
Dependent	IN	Boolean
ParentFlexValue	IN	varchar2
LongList	IN	Boolean
SegmentValue	IN	varchar2
RunProductCode	OUT	Boolean
Column1ValueList	OUT	mv_list_util.MV_Varchar_List_T
Column2ValueList	OUT	mv_list_util.MV_Varchar_List_T

- **PopulateSegmentForLine**: Populates the DFF segment.

Parameters for the PopulateSegmentForLine Procedure		
Parameter	**Input/Output**	**Type**
LineData	IN	mverp_apinvoice.Distribution_R,
ERPUserID	IN	varchar2
ResponsibilityID	IN	number
ApplicationID	IN	number
DescrFlexSegment	IN	MVERP_DFM_Descr_Flex. DescrFlexSegment_R
DescrFlexContextCode	IN	varchar2
Dependent	IN	Boolean
ParentFlexValue	IN	varchar2
LongList	IN	Boolean
SegmentValue	IN	varchar2
RunProductCode	OUT	Boolean
Column1ValueList	OUT	mv_list_util.MV_Varchar_List_T
Column2ValueList	OUT	mv_list_util.MV_Varchar_List_T

- **SetContextValue**: Returns the descriptive flex field (DFF) context for a distribution line. This is only relevant when the DFM is created with DFM_FIELD_DFF field enabled. (Oracle R12 environments only)

Parameters for the SetContextValue Function		
Parameter	**Input/Output**	**Type**
InvoiceData	IN	MVERP_APInvoice.Header_R
DistributionData	IN	MVERP_APInvoice.Distribution_R
RequireSubFormYN	OUT	varchar2
RunProductCode	OUT	Boolean
IsValid	OUT	Boolean
ErrorMessage	OUT	varchar2
return		varchar2

- **SetContextValueForLine**: Returns the descriptive flex field (DFF) context for an invoice line. This is only relevant when the DFM is created with DFM_FIELD_DFF field enabled. (Oracle R12 environments only)

Parameters for the SetContextValueForLine Function		
Parameter	**Input/Output**	**Type**
InvoiceData	IN	MVERP_APInvoice.Header_R
LineData	IN	MVERP_APInvoice.Distribution_R
RequireSubFormYN	OUT	varchar2
RunProductCode	OUT	Boolean
IsValid	OUT	Boolean
ErrorMessage	OUT	varchar2
return		varchar2

MVERP_DFM_LOV_CUSTOM

Previously named "MVOA_DFM_LOV_CUSTOM"

Procedures and functions from this package are called if ENABLE_MVERP_DFM_LOV_CUSTOM preference is set to TRUE.

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body and each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **GetValidExpenditureOrgs**: Populates a custom LOV for the Expenditure Organization ACD form control. This is only relevant when the DFM is created with UseProjects set to TRUE.

Parameters for the GetValidExpenditureOrgs Procedure		
Parameter	**Input/Output**	**Type**
ERPUserID	IN	varchar2
ResponsabilityID	IN	number
ApplicationID	IN	number
OrganizationName	IN	varchar2
OrganizationNamePattern	IN	varchar2
OrganizationID	IN	number
ProjectID	IN	number
ExpenditureItemDate	IN	date
ListCount	OUT	number
OrganizationNameList	OUT	MVERP_Util.Varchar2List_T
OrganizationIDList	OUT	MVERP_Util.NumberList_T
RunProductCode	OUT	Boolean

- **GetValidExpenditureTypes**: Populates a custom LOV for the Expenditure Type ACD form control. This is only relevant when the DFM is created with UseProjects set to TRUE.

Parameters for the GetValidExpenditureTypes Procedure		
Parameter	**Input/Output**	**Type**
ERPUserID	IN	varchar2
ResponsabilityID	IN	number
ApplicationID	IN	number
InvoiceID	IN	varchar2
ProjectID	IN	number
TaskID	IN	number
ExpenditureItemDate	IN	date
ExpenditureType	IN	varchar2
ListCount	OUT	number
ExpenditureTypeList	OUT	MVERP_Util.Varchar2List_T
DescriptionList	OUT	MVERP_Util.Varchar2List_T
UnitOfMeasureList	OUT	MVERP_Util.Varchar2List_T
RunProductCode	OUT	Boolean

- **GetValidProjects**: Populates a custom LOV for the Project ACD form control. This is only relevant when the DFM is created with UseProjects set to TRUE.

Parameters for the GetValidProjects Procedure		
Parameter	**Input/Output**	**Type**
ERPUserID	IN	varchar2
ResponsabilityID	IN	number
ApplicationID	IN	number
ProjectID	IN	number, default NULL
ProjectNumber	IN	varchar2, default NULL
ProjectNumberPattern	IN	varchar2, default NULL
ListCount	OUT	number
ProjectNumberList	OUT	MVERP_Util.Varchar2List_T
ProjectNameList	OUT	MVERP_Util.Varchar2List_T
StartDateList	OUT	MVERP_Util.DateList_T
CompletionDateList	OUT	MVERP_Util.DateList_T
ProjectIDList	OUT	MVERP_Util.NumberList_T
RunProductCode	OUT	Boolean

- **GetValidTasks**: Populates a custom LOV for the Task ACD form control. This is only relevant when the DFM is created with UseProjects set to TRUE.

Parameters for the GetValidTasks Procedure		
Parameter	**Input/Output**	**Type**
ERPUserID	IN	varchar2
ResponsabilityID	IN	number
ApplicationID	IN	number
ProjectID	IN	number
TaskNumber	IN	varchar2
TaskNumberPattern	IN	varchar2
TaskID	IN	number
ListCount	OUT	number
TaskNumberList	OUT	MVERP_Util.Varchar2List_T
IndentedTaskNameList	OUT	MVERP_Util.Varchar2List_T
StartDateList	OUT	MVERP_Util.DateList_T
CompletionDateList	OUT	MVERP_Util.DateList_T
ChargableFlagList	OUT	MVERP_Util.Varchar2List_T
TaskIDList	OUT	MVERP_Util.NumberList_T
RunProductCode	OUT	Boolean

- **LongList**: Returns the following
  - TRUE: The LOV query should run
  - FALSE: The LOV query should not run because the criteria are not restrictive enough

Parameters for the LongList Procedure		
Parameter	**Input/Output**	**Type**
FieldName	IN	varchar2
ERPUserID	IN	varchar2
ResponsabilityID	IN	number
ApplicationID	IN	number
Pattern	IN	varchar2
BindControlNameList	IN	MVERP_Util.Varchar2List_T
BindControlValueList	IN	MVERP_Util.Varchar2List_T
ExecuteQuery	OUT	Boolean
RunProductCode	OUT	Boolean

- **PopulateFlexSegment**: Populates the LOV of flex segment values for a distribution line. This is only relevant when the DFM is created with the DFM_FIELD_DFF field enabled.

Parameters for the PopulateFlexSegment Procedure		
Parameter	**Input/Output**	**Type**
DistributionData	IN	MVERP_APInvoice.Distribution_R
SegmentNumber	IN	number
FlexValueSetID	IN	number
Secure	IN	Boolean, default FALSE
ResponsabilityID	IN	number
Dependent	IN	Boolean, default FALSE
ParentFlexValue	IN	varchar2
SegmentValue	IN	varchar2
Column1ValueList	OUT	MVERP_Util.Varchar2List_T
Column2ValueList	OUT	MVERP_Util.Varchar2List_T
RunProductCode	OUT	Boolean

- **PopulateFlexSegmentForLine**: Populates the LOV of flex segment values for an invoice line. This is only relevant when the DFM is created with the DFM_FIELD_DFF field enabled.

Parameters for the PopulateFlexSegmentForLine Procedure		
Parameter	**Input/Output**	**Type**
LineData	IN	MVERP_APInvoice.Distribution_R
SegmentNumber	IN	number

Parameters for the PopulateFlexSegmentForLine Procedure		
Parameter	**Input/Output**	**Type**
FlexValueSetID	IN	number
Secure	IN	Boolean, default FALSE
ResponsabilityID	IN	number
Dependent	IN	Boolean, default FALSE
ParentFlexValue	IN	varchar2
SegmentValue	IN	varchar2
Column1ValueList	OUT	MVERP_Util.Varchar2List_T
Column2ValueList	OUT	MVERP_Util.Varchar2List_T
RunProductCode	OUT	Boolean

MVOA_SSI_CUSTOM

- **CreateInvoiceWorkItem**: Returns the WorkItem Class Name, Workflow Name, and WorkItem Property Name and Values for the work item being created. Do not create a work item within this procedure.

  To run the customer logic, the procedure must set the RunProductCode output parameter to FALSE. Otherwise, product logic runs.

Parameters for the CreateInvoiceWorkItem Procedure		
Parameter	**Input/Output**	**Type**
RequestData	IN	MVOA_SSI_Util.RequestData_T
RunProductCode	OUT	Boolean
WorkItemClassName	OUT	sf_workitem_class.class_name%TYPE
WorkflowName	OUT	sf_workflow.workflow_name%TYPE
WorkItemPropertyNameList	OUT	SF_Client.WorkItemPropertyNameList_T
WorkItemPropertyValueList	OUT	SF_Client.WorkItemPropertyValueList_T

- **PostCreateRenderDocument**: Allows the rendered document to be modified to include DFF or Custom field information in the form of markups.

Parameters for the PostCreateRenderDocument Procedure		
Parameter	**Input/Output**	**Type**
DocumentID	IN	Number
InvoiceID	IN	Number*
AppsUserData	IN	MVOA_Apps.AppsUserData_T
RunProductCode	OUT	Boolean
*Interface Invoice ID		

- **SupplierLongListCheck**: Returns a boolean value indicating whether the suppliers list could be extracted using the search criteria provided.

Parameters for the SupplierLongListCheck Function		
Parameter	**Input/Output**	**Type**
SupplierNameCriteria	IN	Varchar2
SupplierNumberCriteria	IN	Varchar2
return		Boolean

- **GetCustomLOV**: Populates the custom LOV based on the search criteria.

Parameters for the GetCustomLOV Procedure		
Parameter	**Input/Output**	**Type**
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
LineNumber	IN	mvoa_ssi_custom_field_value.line_number%Type Default Null
FieldID	IN	mvoa_ssi_custom_field.field_id%TYPE
DescriptionFilter	IN	Varchar2 DEFAULT null
ValueList	OUT	MVOA_SSI_UI_Invoice_Controls.ValueList_T
DescriptionList	OUT	MVOA_SSI_UI_Invoice_Controls.DescriptionList_T
ListCount	OUT	number
FilteredFlag	IN OUT	Boolean

- **CustomLongListCheck**: Returns a boolean value indicating whether the Custom LOV could be extracted using the search criteria provided.

Parameters for the CustomLongListCheck Function		
Parameter	**Input/Output**	**Type**
FieldID	IN	number
DescriptionCriteria	IN	Varchar2
return		Boolean

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameter for the PackageVersion Function	
Parameter	**Type**
return	varchar2

- **SSICustomExpireAction**: Determines and processes a custom SSI invoice expiration, if one exists.

Parameter for the SSICustomExpireAction Procedure		
Parameter	**Input/ Output**	**Type**
WorkItemInstanceID	IN	sf_workitem_instance. workitem_instance_id %type

MVOA_SSI_DFF_CUSTOM

- **GetDefaultContext**: Populates the default DFF context.

Parameters for the GetDefaultContext Procedure		
Parameter	**Input/Output**	**Type**
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
InvoiceData	IN	MVOA_SSI_INVOICES.InvoiceHeaderRecord_T
DistributionData	IN	MVOA_SSI_INVOICES.InvoiceDistributionRecord_T
AccountingFlexValues	IN	MVOA_SSI_REQUESTS.SegmentValueList_T
LineNumber	IN	mvoa_ssi_invoice_line.line_number%type Default NULL
ContextCodeList	IN	mvoa_ssi_ui_dff.ContextCodeList_T
DefaultValue	IN OUT	Varchar2

- **GetSegmentInfo**: Populates the segment list using the search criteria provided.

Parameters for the GetSegmentInfo Procedure		
Parameter	**Input/Output**	**Type**
UserID	IN	mv_user_profile.user_id%TYPE
RequestID	IN	mvoa_ssi_request.request_id%TYPE
LineNumber	IN	mvoa_ssi_invoice_line.line_number%type Default NULL
ContextCode	IN	varchar2
SegmentInfo	IN	MVOA_Descriptive_Flex.DescrFlexSegment_T
newSegmentInfo	IN OUT	MVOA_SSI_UI_DFF.Segment_T

- **SegmentLongListCheck**: Returns a boolean value indicating whether the segment list could be extracted using the search criteria provided.

Parameters for the SegmentLongListCheck Function		
Parameter	**Input/Output**	**Type**
ContextCode	IN	varchar2
SegmentNumber	IN	Number
ValueCriteria	IN	Varchar2
DescriptionCriteria	IN	Varchar
return		Boolean

- **GetSegmentValues**: Populates segment values.

Parameters for the GetSegmentValues Procedure		
Parameter	**Input/ Output**	**Type**
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
InvoiceData	IN	MVOA_SSI_INVOICES.InvoiceHeaderRecord_T
DistributionData	IN	MVOA_SSI_INVOICES.InvoiceDistributionRecord_T
AccountingFlexValues	IN	MVOA_SSI_REQUESTS.SegmentValueList_T
LineNumber	IN	mvoa_ssi_invoice_line.line_number%type DEFAULT NULL
ContextCode	IN	varchar2
SegmentNumber	IN	Number
ParentFlexValue	IN	varchar2 default NULL
SearchCriteria	IN	Varchar2
SearchCriteria2	IN	Varchar2
FlexValueList	IN OUT	MVERP_Util.Varchar2List_T
FlexValueDescriptionList	IN OUT	MVERP_Util.Varchar2List_T

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameters for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVOA_SSI_INV_RENDER_CUSTOM

- **RenderInvoice**: Renders the invoice.

Parameters for the RenderInvoice Procedure		
Parameter	**Input/Output**	**Type**
AppsUserData	IN	MVOA_Apps.AppsUserData_T
InvoiceID	IN	Number
SupplierMaintenance	IN	Boolean
DocumentID	IN OUT	Number
DocumentDescription	IN OUT	Varchar2

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameters for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVOA_SSI_REQUESTS_CUSTOM

- **RequestPreSubmit**: Indicates that an invoice and invoice lines were updated with the submitter's information.

Parameters for the RequestPreSubmit Procedure		
Parameter	**Input/Output**	**Type**
RequestData	IN	MVOA_SSI_Util.RequestData_T

- **ValidateInvoiceTotal**: Validates the invoice total.

Parameters for the ValidateInvoiceTotal Function		
Parameter	**Input/Output**	**Type**
RequestID	IN	MVOA_SSI_REQUEST.request_id%type
InvoiceID	IN	MVOA_SSI_REQUEST.interface_invoice_id%type
InvoiceTotal	IN	Number
DistributionTotal	IN	Number
DefaultValue	IN	Boolean
return		Boolean

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameters for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVOA_SSI_UI_INV_CTRL_CUSTOM

- **SupplierLOV**: Populates the ShowEmployees field with a boolean value indicating whether the supplier LOV should be filtered with the Employee look-up type.

Parameters for the SupplierLOV Procedure		
Parameter	**Input/Output**	**Type**
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
ShowEmployees	IN OUT	Boolean

- **InvoiceTypeLookupCode**: Populates the default invoice type look-up code.

Parameters for the InvoiceTypeLookupCode Procedure		
Parameter	**Input/Output**	**Type**
UserID	IN	mv_user_profile.user_id%TYPE

Parameters for the InvoiceTypeLookupCode Procedure		
Parameter	**Input/Output**	**Type**
RequestData	IN	MVOA_SSI_Util.RequestData_T
DefaultValue	IN OUT	varchar2

- **TaxCodeLongListCheck**: Returns a boolean value indicating whether the tax code list could be extracted using the search criteria provided.

Parameters for the TaxCodeLongListCheck Function		
Parameter	**Input/Output**	**Type**
TaxCodeCriteria	IN	Varchar2
return		Boolean

- **ProjectLongListCheck**: Returns a boolean value indicating whether the project list could be extracted using the search criteria provided.

Parameters for the ProjectLongListCheck Function		
Parameter	**Input/Output**	**Type**
ProjectNumberCriteria	IN	Varchar2
ProjectNameCriteria	IN	Varchar2
return		Boolean

- **TaskLongListCheck**: Returns a boolean value indicating whether the task list could be extracted using the search criteria provided.

Parameters for the TaskLongListCheck Function		
Parameter	**Input/Output**	**Type**
TaskNumberCriteria	IN	Varchar2
return		Boolean

- **ExpOrgLongListCheck**: Returns a boolean value indicating whether the expenditure organization list could be extracted using the search criteria provided.

Parameters for the ExpOrgLongListCheck Function		
Parameter	**Input/Output**	**Type**
OrganizationCriteria	IN	Varchar2
return		Boolean

- **ExpTypeLongListCheck**: Returns a boolean value indicating whether the expenditure type list could be extracted using the search criteria provided.

Parameters for the ExpTypeLongListCheck Function		
Parameter	**Input/Output**	**Type**
ValueCriteria	IN	Varchar2
return		Boolean

- **AwardLongListCheck**: Returns a boolean value indicating whether the award list could be extracted using the search criteria provided.

Parameters for the AwardLongListCheck Function		
Parameter	**Input/Output**	**Type**
AwardNumberCriteria	IN	Varchar2
return		Boolean

- **AccountingFlexLongListCheck**: Returns a boolean value indicating whether the accounting flex list could be extracted using the search criteria provided.

Parameters for the AccountingFlexLongListCheck Function		
Parameter	**Input/Output**	**Type**
SegmentNumber	IN	Number
ValueCriteria	IN	Varchar2
DescriptionCriteria	IN	Varchar2
return		Boolean

- **AddressLine2**: Populates the ShowAddressLine2 field with a boolean value indicating whether the address line should appear in the interface.

Parameters for the AddressLine2 Procedure		
Parameter	**Input/Output**	**Type**
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
ShowAddressLine2	IN OUT	Boolean

- **AccountingFlexDfltValueList**: Populates the default Accounting Flex value list. To hide Accounting Flex, use the standard field join point.

Parameters for the AccountingFlexDfltValueList Procedure		
Parameter	**Input/Output**	**Type**
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
InvoiceHeaderData	IN	MVOA_SSI_Invoices.InvoiceHeaderRecord_T
InvoiceDistributionData	IN	MVOA_SSI_Invoices.InvoiceDistributionRecord_T
DefaultValueList	IN OUT	MVOA_SSI_UI_Invoice_Controls.ValueList_T

- **State**: Populates the ShowState field with a boolean value indicating whether the state should appear in the interface.

Parameters for the State Procedure		
Parameter	**Input/Output**	**Type**
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T

Parameters for the State Procedure		
Parameter	**Input/Output**	**Type**
ShowState	IN OUT	Boolean

- **ZipCode**: Populates the ShowZipCode field with a boolean value indicating whether the zip code appears in the interface.

Parameters for the ZipCode Procedure		
Parameter	**Input/Output**	**Type**
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
ShowZipCode	IN OUT	Boolean
FieldLabel	IN OUT	varchar2

- **Province**: Populates the Province field with a boolean value indicating whether the province should appear in the interface.

Parameters for the Province Procedure		
Parameter	**Input/Output**	**Type**
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
ShowProvince	IN OUT	Boolean

- **OrganizationID**: Populates the list of organizations.

Parameters for the OrganizationID Procedure		
Parameter	**Input/Output**	**Type**
UserID	IN	mv_user_profile.user_id%TYPE
OrgInstanceID	IN	mvt_workflow_utility.OrgInstanceIDList_T
OrgID	IN	mvt_workflow_utility.OrgIDList_T
OrgName	IN	mvt_workflow_utility.OrgNameList_T
OrgShortName	IN	mvt_workflow_utility.OrgShortNameList_T
ListLength	IN	number
IndexList	OUT	mvoa_ssi_ui_invoice_controls.indexlist_t
IndexListCount	OUT	number

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameters for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVOA_SSI_UI_INVOICES_CUSTOM

- **PostSetDistribution**: Indicates that the distribution data is set.

Parameters for the PostSetDistribution Procedure		
Parameter	**Input/ Output**	**Type**
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
InvoiceHeaderData	IN	MVOA_SSI_Invoices.InvoiceHeaderRecord_T
InvoiceDistributionData	IN	MVOA_SSI_Invoices.InvoiceDistributionRecord_T
AccountingSegmentValueList	IN	MVOA_SSI_Requests.SegmentValueList_T
ValidateDistribution	IN	Boolean

- **PreDeleteDistribution**: Indicates that the distribution will be deleted.

Parameters for the PreDeleteDistribution Procedure		
Parameter	**Input/ Output**	**Type**
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
InvoiceHeaderData	IN	MVOA_SSI_Invoices.InvoiceHeaderRecord_T
InvoiceDistributionData	IN	MVOA_SSI_Invoices.InvoiceDistributionRecord_T
AccountingSegmentValueList	IN	MVOA_SSI_Requests.SegmentValueList_T

- **AccountingFlexLOV**: Filters the Accounting Flex LOV by rebuilding the IndexList. Instead of using this option, Kofax recommends that you implement the Oracle Responsibility that restricts these values.

  If you restrict a dependent set, a user might not have any values available.

Parameters for the AccountingFlexLOV Procedure		
Parameter	**Input/ Output**	**Type**
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
InvoiceHeaderData	IN	MVOA_SSI_Invoices.InvoiceHeaderRecord_T
InvoiceDistributionData	IN	MVOA_SSI_Invoices.InvoiceDistributionRecord_T
AccountingSegmentValueList	IN	MVOA_SSI_Requests.SegmentValueList_T
SegmentNumber	IN	number
ValueList	IN	MVOA_SSI_UI_Invoice_Controls.ValueList_T
DescriptionList	IN	MVOA_SSI_UI_Invoice_Controls.DescriptionList_T
ListCount	IN	number
DefaultValue	IN	varchar2
IndexList	OUT	IndexList_T
IndexListCount	IN OUT	number
DefaultValueIndex	IN OUT	number

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameters for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVOA_SSI_UI_Templates_Custom

- **OverrideCustomField**: Populates the custom field value.

Parameters for the OverrideCustomField Procedure		
Parameter	**Input/Output**	**Type**
UserID	IN	varchar2
DocumentTypeID	IN	Number
LineNumber	IN	mvoa_ssi_custom_field_value.line_number%TYPE
FieldID	IN	mvoa_ssi_custom_field_value.field_id%TYPE
FieldLabel	IN	mvoa_ssi_custom_field.field_label%TYPE
FieldValue	IN OUT	mvoa_ssi_custom_field_value.field_value%TYPE

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameters for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVRM_ACTION_CUSTOM

Parameters for the GetBatchActionTypeList Procedure		
Parameter	**Input/Output**	**Type**
UserID	IN	mv_user_profile.user_id%type
WorkstationSerialNumber	IN	mv_preference.default_value%type
BatchActionTypeList	IN OUT	MVRM_Action.BatchActionTypeList_T
BatchActionAttributeList	IN OUT	MVRM_Action.BatchActionAttributeList_T
DocumentActionDataList	IN OUT	MVRM_Action.DocumentActionDataList_T

Parameters for the GetBatchActionData Procedure		
Parameter	**Input/ Output**	**Type**
BatchActionData	IN OUT	MVRM_Action.BatchActionData_T
BatchActionAttributeList	IN OUT	MVRM_Action.BatchActionAttributeList_T
DocumentActionDataList	IN OUT	MVRM_Action.DocumentActionDataList_T

Parameters for the GetBatchActionMessage Procedure		
Parameter	**Input/Output**	**Type**
BatchActionTypeName	IN	varchar2
PageTitle	IN OUT	varchar2
NumMarkviewDocs	IN	number
NumNonMarkviewDocs	IN	number
Message	IN OUT	varchar2
Color	IN OUT	varchar2
IsError	IN OUT	Boolean

Parameters for the SubmitAction Procedure		
Parameter	**Input/Output**	**Type**
BatchActionData	IN OUT	MVRM_Action.BatchActionData_T
BatchActionAttributeList	IN OUT	MVRM_Action.BatchActionAttributeList_T
DocumentActionDataList	IN OUT	MVRM_Action.DocumentActionDataList_T
StatusMessage		varchar2
ErrorMessage		varchar2
Result		Boolean

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameter for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameter for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVRM_BAR_CODE_CUSTOM

Use this package to customize the bar code cover page associated with each object and accessible from the Bar Code button. A default implementation is supplied.

Parameters for the CustomBarcodePage Procedure	
Parameter	**Description**
ObjectID	the unique id of the object
ObjectClassName	the name of the object class
QualifiedObjectName	the complete, fully qualified object name
ErrorMessage	If an error occurs within the procedure, use the ErrorMessage parameter to return a meaningful message to the end user.
Result	If an error occurs within the procedure, set Result to FALSE. A Result of TRUE indicates normal, successful completion.

Table Name	Index
type Varchar2Array is table of Varchar2(200)	index by binary_integer

CustomBarcodePage Procedure Information		
Parameter	**Input/Output**	**Type**
ObjectID	IN	mvrm_object.Object_ID%type
ObjectClassName	IN	mvrm_object_class.Object_Class_Name%type
QualifiedObjectName	IN	varchar2
ErrorMessage	OUT	varchar2
Result	OUT	Boolean

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

GetPackageVersion Procedure Information		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

PackageVersion Function Information	
Parameter	**Type**
return	varchar2

MVRM_DISPOSITION_CUSTOM

Parameter for the GetDispositionActions Function	
Parameter	**Type**
return	mvrm_util.Varchar2List_T

Parameters for the DispositionCustom Procedure		
Parameter	**Input/Output**	**Type**
obj	IN	mvrm_util.Object_R
ObjDisp	IN	mvrm_util.ObjectDispositionList_T

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameter for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVRM_OBJECT_CUSTOM

Use this package to perform operations prior to object creation. PreCreateObject is called prior to the creation of an object when the object is created through the UI or the MVRM_Object_Util.CreateNewObject API. Use this procedure to perform validation specific to object creation (for example, to prevent a user from creating an object) ParentObjectID: The Object ID of the created object parent. May be NULL if no container is specified for the new object.

Parameter	Description
ObjectClassID	The Object Class ID for the new object
Name	The Name of the new object
DocumentID	The Document ID associated with the new object; may be NULL if no Document is specified for the new object
UserID	User ID of the user creating the new object
PropertyList(i).PropertyName	A list of property names to set for the new object
PropertyList(i).NewPropertyValue	A list of the associated property values
Status	Denotes whether the created object is filed. Use one of MVRM_Object_Util.MVRM_STATUS_NOTFILED or MVRM_Object_Util.MVRM_STATUS_FILED
ErrorMessage	If an error occurs in the procedure, use the ErrorMessage parameter to return a meaningful message to the user
Result	If an error occurs within the procedure, set Result to FALSE; TRUE indicates normal, successful completion

- **PreCreateObject**: Called after the creation of an object through the user interface or the MVRM_Object_Util.CreateNewObject API. Use this procedure to perform additional operations following the creation of an object (for example, to give a group of users privileges to edit the new object).

Parameters for the PreCreateObject Procedure	
Parameter	**Description**
ParentObjectID	The Object ID of the parent of the object. Can be NULL if no container is specified for the new object
ObjectClassID	The Object Class ID for the new object
Name	The Name of the new object
DocumentID	The Document ID associated with the new object. Can be NULL if no Document is specified for the new object
UserID	User ID of the user creating the new object
PropertyList(i).PropertyName	A list of property names to set for the new object
PropertyList(i).NewPropertyValue	A list of the associated property values
Status	Denotes whether the created object is filed. Use one of MVRM_Object_Util.MVRM_STATUS_NOTFILED or MVRM_Object_Util.MVRM_STATUS_FILED
ErrorMessage	If an error occurs within the procedure, use the ErrorMessage parameter to return a meaningful message to the user
ObjectID	The ID of the newly created object
Result	If an error occurs within the procedure, set Result to FALSE; TRUE indicates normal, successful completion

PreCreateObject Procedure Information		
Parameter	**Input/Output**	**Type**
ParentObjectID	IN	mvrm_object.object_id%TYPE
ObjectClassID	IN	mvrm_object_class.object_class_id%TYPE
Name	IN	mvrm_object_property.property_value%TYPE
DocumentID	IN	mvrm_object_version.document_id%TYPE
UserID	IN	mv_user_profile.user_id%TYPE
PropertyList	IN OUT	mvrm_util.ObjectPropertyList_T
Status	IN	mvrm_object_version.object_status%TYPE
ErrorMessage	OUT	Varchar2
Result	OUT	Boolean

- **PostUpload**: Called after a document is uploaded through the interface. Use this procedure to perform additional operations following the document upload.

Parameters for the PostUpload Procedure	
Parameter	**Description**
ObjectClassID	The Object Class ID associated with the document
Name	The Name of the object created along with the document
DocumentID	The Document ID of the uploaded document
UserID	User ID of the user creating the new object
PropertyList(i).PropertyName	A list of property names to set for the object
PropertyList(i).NewPropertyValue	A list of the associated property values
Status	Denotes whether the object associated with the uploaded document is filed. Use one of MVRM_Object_Util.MVRM_STATUS_NOTFILED or MVRM_Object_Util.MVRM_STATUS_FILED
ErrorMessage	If an error occurs within the procedure, use the ErrorMessage parameter to return a meaningful message to the user
Result	If an error occurs within the procedure, set Result to FALSE; TRUE indicates normal, successful completion

PostUpload Procedure Information		
Parameter	**Input/Output**	**Type**
ObjectClassID	IN	mvrm_object_class.object_class_id%TYPE
Name	IN	mvrm_object_property.property_value%TYPE
DocumentID	IN	mvrm_object_version.document_id%TYPE
UserID	IN	mv_user_profile.user_id%TYPE
PropertyList	IN OUT	mvrm_util.ObjectPropertyList_T

PostUpload Procedure Information		
Parameter	**Input/Output**	**Type**
Status	IN	mvrm_object_version.object_status%TYPE
ObjectID	IN	mvrm_object.object_id%TYPE
ErrorMessage	OUT	Varchar2
Result	OUT	Boolean

- **PreUpdateObject**: Called before a document is updated through the interface. Use this procedure to perform additional operations following the update, for example, to prevent a user from updating an object.

Parameters for the PreUpdateObject Procedure	
Parameter	**Description**
ObjectID	The Object ID that was updated
ObjectClassID	The Object Class ID of the updated object
Name	The Name of the updated object
DocumentID	The Document ID associated with the object
UserID	User ID of the user who updated the object
PropertyList(i).PropertyName	A list of updated property names to set for the object
PropertyList(i).NewPropertyValue	A list of the associated property values
Status	Denotes whether the object associated with the uploaded document is filed. Use one of MVRM_Object_Util.MVRM_STATUS_NOTFILED or MVRM_Object_Util.MVRM_STATUS_FILED
ErrorMessage	If an error occurs within the procedure, use the ErrorMessage parameter to return a meaningful message to the user
Result	If an error occurs within the procedure, set Result to FALSE; TRUE indicates normal, successful completion

PreUpdateObject Procedure Information		
Parameter	**Input/Output**	**Type**
ParentObjectID	IN OUT	mvrm_object.object_id%TYPE
ObjectClassID	IN	mvrm_object_class.object_class_id%TYPE
Name	IN	mvrm_object_property.property_value%TYPE
DocumentID	IN	mvrm_object_version.document_id%TYPE
UserID	IN	mv_user_profile.user_id%TYPE
PropertyList	IN OUT	mvrm_util.ObjectPropertyList_T
Status	IN	mvrm_object_version.object_status%TYPE
ObjectID	IN	mvrm_object.object_id%TYPE

PreUpdateObject Procedure Information		
Parameter	**Input/Output**	**Type**
ErrorMessage	OUT	Varchar2
Result	OUT	Boolean

- **PostUpdateObject**: Called after a document is updated through the interface. Use this procedure to perform additional operations following the update, for example, to cascade property values to the children of an object.

Parameters for the PostUpdateObject Procedure	
Parameter	**Description**
ObjectID	The Object ID that was updated
ObjectClassID	The Object Class ID of the updated object
Name	The Name of the updated object
DocumentID	The Document ID associated with the object
UserID	User ID of the user who updated the object
PropertyList(i).PropertyName	A list of updated property names to set for the object
PropertyList(i).NewPropertyValue	A list of the associated property values
Status	Denotes whether the object associated with the uploaded document is filed. Use one of MVRM_Object_Util.MVRM_STATUS_NOTFILED or MVRM_Object_Util.MVRM_STATUS_FILED
ErrorMessage	If an error occurs within the procedure, use the ErrorMessage parameter to return a meaningful message to the user
Result	If an error occurs within the procedure, set Result to FALSE; TRUE indicates normal, successful completion

PostUpdateObject Procedure Information		
Parameter	**Input/Output**	**Type**
ObjectID	IN OUT	mvrm_object.object_id%TYPE
ObjectClassID	IN	mvrm_object_class.object_class_id%TYPE
Name	IN	mvrm_object_property.property_value%TYPE
DocumentID	IN	mvrm_object_version.document_id%TYPE
UserID	IN	mv_user_profile.user_id%TYPE
PropertyList	IN OUT	mvrm_util.ObjectPropertyList_T
Status	IN	mvrm_object_version.object_status%TYPE
ErrorMessage	OUT	Varchar2
Result	OUT	Boolean

- **PreAttachDocumentToObject**: Called before associating a document with an object when the document is associated through the interface or by using the MVOA_Object_Util.AssociateDocument API. Use this procedure to perform additional operations prior to associating the document, for example, to prevent a user from associating a document.

Parameters for the PreAttachDocumentToObject Procedure	
Parameter	**Description**
AttachObjectID	The Object ID to which the document is associated
DocumentID	The Document ID to associate with the object
DocObjectID	An Object ID that has a document to re-associate with the AttachObjectID
Filename	The document filename
ErrorMessage	If an error occurs within the procedure, use the ErrorMessage parameter to return a meaningful message to the user
Result	If an error occurs within the procedure, set Result to FALSE; TRUE indicates normal, successful completion

PreAttachDocumentToObject Procedure Information		
Parameter	**Input/Output**	**Type**
AttachObjectID	IN	mvrm_object.object_id%TYPE
DocumentID	IN	mvrm_object_version.document_id%TYPE default null
DocObjectID	IN	mvrm_object.object_id%TYPE default null
Filename	IN	mvrm_object_property.property_value%TYPE default null
ErrorMessage	OUT	Varchar2
Result	OUT	Boolean

- **PostAttachDocumentToObject**: Performs operations before removing a document from an object.

Parameters for the PostAttachDocumentToObject Procedure	
Parameter	**Description**
AttachObjectID	The Object ID to which the document is associated
DocumentID	The Document ID to associate with the object
DocObjectID	An Object ID that has a document to re-associate with the AttachObjectID
Filename	The document filename
ErrorMessage	If an error occurs within the procedure, use the ErrorMessage parameter to return a meaningful message to the user
Result	If an error occurs within the procedure, set Result to FALSE; TRUE indicates normal, successful completion

PostAttachDocumentToObject Procedure Information		
Parameter	**Input/Output**	**Type**
AttachObjectID	IN	mvrm_object.object_id%TYPE
DocumentID	IN	mvrm_object_version.document_id%TYPE default null
DocObjectID	IN	mvrm_object.object_id%TYPE default null
Filename	IN	mvrm_object_property.property_value%TYPE default null
ErrorMessage	OUT	Varchar2
Result	OUT	Boolean

- **PreRemoveDocumentAssociation**: Called before removing a document from an object when the document is removed through the interface or by using the MVOA_Object_Util.RemoveDocument API. Use this procedure to perform additional operations before removing the document.

Parameters for the PreRemoveDocumentAssociation Procedure	
Parameter	**Description**
ObjectID	The Object ID from which the document was removed
ErrorMessage	If an error occurs within the procedure, use the ErrorMessage parameter to return a meaningful message to the user
Result	If an error occurs within the procedure, set Result to FALSE; TRUE indicates normal, successful completion

PreRemoveDocumentAssociation Procedure Information		
Parameter	**Input/Output**	**Type**
ObjectID	IN	mvrm_object.object_id%TYPE
ErrorMessage	OUT	Varchar2
Result	OUT	Boolean

- **PostRemoveDocumentAssociation**: Performs operations after removing a document from an object.

PostRemoveDocumentAssociation Procedure Information		
Parameter	**Input/Output**	**Type**
ObjectID	IN	mvrm_object.object_id%TYPE
ErrorMessage	OUT	Varchar2
Result	OUT	Boolean

- **PreDeleteObject**: Called before deleting an object when the object is deleted through the interface or by using the MVOA_Object_Util.DeleteObject API. Use this procedure to perform additional operations before deleting the object.

Parameters for the PreDeleteObject Procedure	
Parameter	**Description**
ObjectID	The Object ID being deleted
DeleteMetadata	TRUE if the object metadata is being deleted along with associated documents
ErrorMessage	If an error occurs within the procedure, use the ErrorMessage parameter to return a meaningful message to the user
Result	If an error occurs within the procedure, set Result to FALSE; TRUE indicates normal, successful completion

PreDeleteObject Procedure Information		
Parameter	**Input/Output**	**Type**
ObjectID	IN	mvrm_object.object_id%TYPE
DeleteMetadata	IN	Varchar2
ErrorMessage	OUT	Varchar2
Result	OUT	Boolean

- **PostDeleteObject**: Called after deleting an object when the object is deleted through the interface or by using the MVOA_Object_Util.DeleteObject API. Use this procedure to perform additional operations after deleting the object.

Parameters for the PostDeleteObject Procedure	
Parameter	**Description**
ObjectID	The Object ID being deleted
DeleteMetadata	TRUE if the object metadata is being deleted along with associated documents
ErrorMessage	If an error occurs within the procedure, use the ErrorMessage parameter to return a meaningful message to the user
Result	If an error occurs within the procedure, set Result to FALSE; TRUE indicates normal, successful completion

PostDeleteObject Procedure Information		
Parameter	**Input/Output**	**Type**
ObjectID	IN	mvrm_object.object_id%TYPE
DeleteMetadata	IN	Varchar2
ErrorMessage	OUT	Varchar2
Result	OUT	Boolean

- **PreCheckoutObject**: Called before checking out an object (specifically when an object is checked out through the interface). Use this procedure to perform additional operations before issuing a checkout on the object.

Parameters for the PreCheckoutObject Procedure	
Parameter	**Description**
ObjectID	The Object ID being checked out
UserID	The User ID issuing the check out
ErrorMessage	If an error occurs within the procedure, use the ErrorMessage parameter to return a meaningful message to the user
Result	If an error occurs within the procedure, set Result to FALSE; TRUE indicates normal, successful completion

PreCheckoutObject Procedure Information		
Parameter	**Input/Output**	**Type**
ObjectID	IN	mvrm_object.object_id%TYPE
UserID	IN	Varchar2
RollbackFlag	IN	boolean default FALSE
ErrorMessage	OUT	Varchar2
Result	OUT	Boolean

- **PostCheckoutObject**: Called after checking out an object when the object is checked out through the interface. Use this procedure to perform additional operations after issuing a checkout on the object.

Parameters for the PostCheckoutObject Procedure	
Parameter	**Description**
ObjectID	The Object ID being checked out
UserID	The User ID issuing the checkout
ErrorMessage	If an error occurs within the procedure, use the ErrorMessage parameter to return a meaningful message to the user
Result	If an error occurs within the procedure, set Result to FALSE; TRUE indicates normal, successful completion

PostCheckoutObject Procedure Information		
Parameter	**Input/Output**	**Type**
ObjectID	IN	mvrm_object.object_id%TYPE
UserID	IN	Varchar2
RollbackFlag	IN	boolean default FALSE
ErrorMessage	OUT	Varchar2
Result	OUT	Boolean

- **PreCheckInObject**: Called before checking in an object when the object is checked in through the interface. Use this procedure to perform additional operations before issuing a check in on the object.

Parameters for the PreCheckInObject Procedure	
Parameter	**Description**
ObjectID	The Object ID being checked in
UserID	The User ID issuing the check in
ErrorMessage	If an error occurs within the procedure, use the ErrorMessage parameter to return a meaningful message to the user
Result	If an error occurs within the procedure, set Result to FALSE; TRUE indicates normal, successful completion

PreCheckInObject Procedure Information		
Parameter	**Input/Output**	**Type**
ObjectID	IN	mvrm_object.object_id%TYPE
UserID	IN	Varchar2
RollbackFlag	IN	boolean default FALSE
ErrorMessage	OUT	Varchar2
Result	OUT	Boolean

- **PostCheckInObject**: Called after checking in an object when the object is checked out through the interface. Use this procedure to perform additional operations after issuing a check in on the object.

Parameters for the PostCheckInObject Procedure	
Parameter	**Description**
ObjectID	The Object ID being checked in
UserID	The User ID issuing the check in
ErrorMessage	If an error occurs within the procedure, use the ErrorMessage parameter to return a meaningful message to the user
Result	If an error occurs within the procedure, set Result to FALSE; TRUE indicates normal, successful completion

PostCheckInObject Procedure Information		
Parameter	**Input/Output**	**Type**
ObjectID	IN	mvrm_object.object_id%TYPE
UserID	IN	Varchar2
RollbackFlag	IN	boolean default FALSE
ErrorMessage	OUT	Varchar2
Result	OUT	Boolean

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameter for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVRM_PRINT_CUSTOM

- **GetDefaultSettings**

Parameters for the GetDefaultSettings Procedure		
Parameter	**Input/Output**	**Type**
BatchPrintData	IN OUT	mvrm_print.BatchPrintData_T
DocumentPrintDataList	IN OUT	mvrm_print.DocumentPrintDataList_T

- **SubmitPrintRequest**

Parameters for the SubmitPrintRequest Procedure		
Parameter	**Input/Output**	**Type**
BatchPrintData	IN OUT	mvrm_print.BatchPrintData_T
DocumentPrintDataList	IN OUT	mvrm_print.DocumentPrintDataList_T
ErrorMessage	OUT	varchar2
Result	OUT	boolean

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

Parameters for the GetPackageVersion Procedure		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

Parameters for the PackageVersion Function	
Parameter	**Type**
return	varchar2

MVRM_SEARCH_CUSTOM

Constant	Value
PAREN_OPEN	constant Varchar2(1) := '('
PAREN_CLOSED	constant Varchar2(1) := ')'
PAREN_NONE	constant Varchar2(1) := ' '
RELATION_EQUALS	constant Varchar2(7) := '='
RELATION_NOT_EQUALS	constant Varchar2(7) := '<>'
RELATION_LESS_THAN	constant Varchar2(7) := '<'
RELATION_GREATER_THAN	constant Varchar2(7) := '>'
RELATION_IS_NULL	constant Varchar2(7) := 'IS NULL'
CONCAT_AND	constant Varchar2(7) := 'AND'
CONCAT_AND_NOT	constant Varchar2(7) := 'AND NOT'
CONCAT_OR	constant Varchar2(7) := 'OR'
CONCAT_NONE	constant Varchar2(7) := ' '

Types Used for Search

type ObjectPropertyNameList_R is record	
ClassName	mvrm_object_class.object_class_name%type
PropertyID	mvrm_object_class_property.object_class_property_id%type
PropertyName	mvrm_object_class_property.property_name%type
type ObjectPropertyNameList_T is table of ObjectPropertyNameList_R index by binary_integer	
type ParameterList_R is record	
ParameterName	varchar2(2000)
ParameterValue	varchar2(2000)
type Parameterlist_T is table of ParameterList_R index by binary_integer	
type StatusList_T is table of varchar2(200) index by binary_integer	
type PropertySearchList_R is record	

IsVisible	Boolean
ParenPrefix	varchar2(1)
PropertyID	Number
Relationship	varchar2(7)
PropertyValue	mvrm_object_property.property_value%type
ParenPostFix	varchar2(1)
Concatenation	varchar2(7)
type PropertySearchList_T is table of PropertySearchList_R index by binary_integer	

Specify additional default search criteria.

This procedure allows additional you to define default search criteria prior to loading the Search page.

Parameter	Description
UserID	The User ID performing the search
FormParameters	A list of the parameters that were passed to the Search page as the result of an HTTP Get or Post
FormParameters(i).ParameterName	A list of parameter names that were passed into the Search page.
FormParameters(i).ParameterValue	A list of the associated parameter values that were passed into the Search page.
PropertyList	A list of object classes and their associated properties
PropertyList(i).PropertyID	The ID of the property
PropertyList(i).PropertyName	The name of the property
PropertyList(i).ClassName	The object class to which this property belongs
StatusList	Not used
PropertySearchList	Specifies the additional search criteria to default on the search page.
PropertySearchList(i).IsVisible	A boolean parameter that denotes whether this search criteria item should be displayed to the user. Set to TRUE if it should be displayed.
PropertySearchList(i).ParenPrefix	The ParenPrefix can be used to group several search criteria together. Use one of the following values: • PAREN_OPEN to begin defining a group • PAREN_NONE for no grouping
PropertySearchList(i).PropertyID	The Property ID to search; maps to the properties defined in the PropertyList parameter.

Parameter	Description
PropertySearchList(i).Relationship	You can use the Relationship to perform a comparison during searching. Use one of the following values: • RELATION_EQUALS: Matches objects that have the given Property Value • RELATION_NOT_EQUALS: Matches object that do not have the given Property Value • RELATION_LESS_THAN: Matches objects with a value less than the given Property Value • RELATION_GREATER_THAN: Matches objects with a value greater than the given Property Value • RELATION_IS_NULL: Matches object with no value for the given Property ID
PropertySearchList(i).PropertyValue	The value for which to search within the specified property
PropertySearchList(i).ParenPostFix	The ParenPrefix can be used to group several search criteria together. Use one of the following values: • PAREN_CLOSED to end a group definition • PAREN_NONE for no grouping
PropertySearchList(i).Concatenation	The Concatenation parameter defines how multiple search criteria should be evaluated. Use one of the following values: • CONCAT_AND denotes that in addition to this criteria being true, all previous search criteria must be true as well. • CONCAT_AND_NOT denotes that this criteria must be false and all previous criteria must be true. • CONCAT_OR denotes that either this criteria or the previous criteria must be true.

- **preInquiryCriteria**: Specifies search criteria in addition to the user's search. This procedure allows search criteria beyond those the user specified to be defaulted prior to calculating the Search Results.

preInquiryCriteria Procedure Information		
Parameter	**Input/Output**	**Type**
userID	IN	mv_user_profile.user_id%type
formParameters	IN	ParameterList_T
propertyList	IN	ObjectPropertyNameList_T
statusList	OUT	StatusList_T
propertySearchList	OUT	PropertySearchList_T

Parameters for the preInquiryCriteria Procedure	
Parameter	**Description**
UserID	Identifies the User ID performing the search

Parameters for the preInquiryCriteria Procedure	
Parameter	**Description**
FormParameters	Provides a list of the parameters that were passed to the Search page as the result of an HTTP Get or Post.
FormParameters(i).ParameterName	Provides a list of parameter names that were passed into the Search page.
FormParameters(i).ParameterValue	Provides a list of the associated parameter values that were passed into the Search page.
PropertyList	Provides a list of object classes and associated properties
PropertyList(i).PropertyID	Identifies the ID of the property
PropertyList(i).PropertyName	Identifies the name of the property
PropertyList(i).ClassName	Specifies the object class to which this property belongs
StatusList	Not used
PropertySearchList	Specifies the additional search criteria to default on the search page.
PropertySearchList(i).IsVisible	Denotes whether to display this search criteria item to the user (boolean parameter). To display the search criteria, set to TRUE.
PropertySearchList(i).ParenPrefix	Defines the start of a grouping of several search criteria. Supported values: • PAREN_OPEN to begin defining a group • PAREN_NONE for no grouping
PropertySearchList(i).PropertyID	Specifies the Property ID to search (maps to the properties defined in the PropertyList parameter).
PropertySearchList(i).Relationship	Performs a comparison during a search. Supported values: • RELATION_EQUALS: Match objects that have the given Property Value • RELATION_NOT_EQUALS: Match objects that do not have the given Property Value • RELATION_LESS_THAN: Match objects with a value less than the given Property Value • RELATION_GREATER_THAN: Match objects with a value greater than the given Property Value • RELATION_IS_NULL: Match objects with no value for the given Property ID
PropertySearchList(i).PropertyValue	Specifies the Value for which to search within the specified property
PropertySearchList(i).ParenPostFix	Defines the end of a grouping of search criteria. Supported values: • PAREN_CLOSED to end the group definition • PAREN_NONE for no grouping

Parameters for the preInquiryCriteria Procedure	
Parameter	**Description**
PropertySearchList(i).Concatenation	Defines how to evaluate multiple search criteria. Supported values: • CONCAT_AND: Denotes that in addition to this criteria being true, all previous search criteria must be true • CONCAT_AND_NOT: Denotes that this criteria must be false, and all previous criteria must be true • CONCAT_OR: Denotes that either this criteria or the previous criteria must be true

- **preQueryCriteria**

preQueryCriteria Procedure Information		
Parameter	**Input/Output**	**Type**
userID	IN	mv_user_profile.user_id%type
formParameters	IN	ParameterList_T
propertyList	IN	ObjectPropertyNameList_T
statusList	OUT	StatusList_T
propertySearchList	OUT	PropertySearchList_T

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

GetPackageVersion Procedure Information		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

PackageVersion Function Information	
Parameter	**Type**
return	varchar2

MVRM_VALID_VALUES_CUSTOM

- **SetValidValues**: Specifies a dynamic list of values for an object property.

SetValidValues Function Information		
Parameter	**Input/Output**	**Type**
ObjectID	IN	mvrm_object.object_id%TYPE
PropertyName	IN	mvrm_object_class_property.property_name%TYPE
UserID	IN	mv_user_profile.user_id%TYPE
ValueFilter	IN	varchar2
ValueFilteredFlag	OUT	boolean
return		mvrm_util.ObjectPropertyValueList_T

Parameter	Description
PropertyName	The name of the Property for which to specify a list of valid values
UserID	The user who is currently logged in

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

GetPackageVersion Procedure Information		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

PackageVersion Function Information	
Parameter	**Type**
return	varchar2

MVT_DOCUMENT_CAPTURE_CUSTOM

This package is called by various document capture routines to provide a centralized location for code used in the creation and initial processing of MarkView documents.

- **PostReadBarCode**: Called once after the bar code is read. Implements standard logic and processing when new documents are created based on the input bar code values.

  If any IN OUT parameters are updated and the UpdateDocument parameter is set to TRUE, MarkView updates the document with the new values before committing.

PostReadBarCode Procedure Information		
Parameter	**Input/ Output**	**Type**
DocumentData	IN OUT	MVT_Document_Capture.DocumentData_T
BarCodeData	IN	MVT_Document_Capture.BarCodeData_T
ScanBatchData	IN	MVT_Document_Capture.ScanBatchData_T
BarCodeSvrData	IN	MVT_Document_Capture.BarCodeSvrData_T
WorkflowName	IN	sf_workflow.workflow_name%TYPE default null
WorkItemPropertyNameList	IN	SF_Client.WorkItemPropertyNameList_T
WorkItemPropertyValueList	IN	SF_Client.WorkItemPropertyValueList_T
UpdateDocument	OUT	boolean
RunProductCode	OUT	boolean

- **PostCreateDocument**: Called once after each newly received document is saved to the database but before it is committed. IN parameters contain information identifying the new document.

  If any IN OUT parameters are updated and the UpdateDocument parameter is set to TRUE, MarkView updates the document with the new values before committing.

PostCreateDocument Procedure Information		
Parameter	**Input/Output**	**Type**
DocumentData	IN OUT	MVT_Document_Capture.DocumentData_T
BarCodeData	IN	MVT_Document_Capture.BarCodeData_T
ScanBatchData	IN	MVT_Document_Capture.ScanBatchData_T
FaxSvrData	IN	MVT_Document_Capture.FaxSvrData_T
BarCodeSvrData	IN	MVT_Document_Capture.BarCodeSvrData_T
WorkflowName	IN	sf_workflow.workflow_name%TYPE default null
WorkItemPropertyNameList	IN OUT	SF_Client.WorkItemPropertyNameList_T
WorkItemPropertyValueList	IN OUT	SF_Client.WorkItemPropertyValueList_T
UpdateDocument	OUT	boolean
RunProductCode	OUT	boolean

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

GetPackageVersion Procedure Information		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number

GetPackageVersion Procedure Information		
Parameter	**Input/Output**	**Type**
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

PackageVersion Function Information	
Parameter	**Type**
return	varchar2

MVT_WEB_DISPLAY_CUSTOM

This package contains custom code used to build and display data in the web pages generated by Web Inquiry and Web Inbox entries in the Web Display tables.

- **GetDocumentList**

GetDocumentList Procedure Information		
Parameter	**Input/Output**	**Type**
PrimaryKey1	IN	Varchar2
PrimaryKey2	IN	Varchar2 default null
PrimaryKey3	IN	Varchar2 default null
PrimaryKey4	IN	Varchar2 default null
PrimaryKey5	IN	Varchar2 default null
EntityName	IN	Varchar2
DocumentID	IN	number default null
UserID	IN	Varchar2 default null
DisplayType	IN	Varchar2 default null
DocumentList	OUT	MVT_Web_Display.Document_List_T
RunProductCode	OUT	Boolean

- **GenerateLOVList**: Creates a selection box with a list of values based on the input information.

GenerateLOVList Procedure Information		
Parameter	**Input/Output**	**Type**
DisplayType	IN	Varchar2
CurrentID	IN	Varchar2
CurrentName	IN	Varchar2
CurrentValue	IN	Varchar2

GenerateLOVList Procedure Information		
Parameter	**Input/Output**	**Type**
CurrentTable	IN	Varchar2
CurrentColumn	IN	Varchar2
LinkedID	IN	Varchar2
LinkedValue	IN	Varchar2
LinkedTable	IN	Varchar2
LinkedColumn	IN	Varchar2
UserID	IN	Varchar2 default null
NameValueItem	OUT	MVT_Web_Display.Lov_List_T
RunProductCode	OUT	Boolean

- **SetDetailVisible**: Determines if a user can view a specified record.

SetDetailVisible Function Information		
Parameter	**Input/Output**	**Type**
UserID	IN	Varchar2
PrimaryKey1	IN	Varchar2
PrimaryKey2	IN	Varchar2
PrimaryKey3	IN	Varchar2
PrimaryKey4	IN	Varchar2
PrimaryKey5	IN	Varchar2
DetailTypeID	IN	number
KeyTable	IN	Varchar2
ErrorMessage	OUT	Varchar2
RunProductCode	OUT	Boolean
Return		Boolean

- **SetSecurity**: Checks the user's security permission against the information stored in the procedure to determine if the user is allowed to run the application. Limits what the user is allowed to view in the result set.

SetSecurity Procedure Information		
Parameter	**Input/Output**	**Type**
DisplayType	IN	Varchar2
UserID	IN	Varchar2
UpdateWhere	IN OUT	Varchar2
IsAuthorized	OUT	Boolean
RunProductCode	OUT	Boolean

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

GetPackageVersion Procedure Information		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

PackageVersion Function Information	
Parameter	**Type**
return	varchar2

MVW_AUTHENTICATE_CUSTOM

This package authenticates web users.

Table	Index
type Varchar2List_T is table of varchar2(32000)	index by binary_integer

- **Authenticate**: Returns TRUE If the UserID and Password combination passed is valid and FALSE if the combination is invalid.

Authenticate Function Information		
Parameter	**Input/Output**	**Type**
UserID	IN	Varchar2
Password	IN OUT	Varchar2
return		Boolean

- **GetSSOUserID**: Retrieves a User ID and (optional) Password using standard HTTP Request mechanisms (Cookie, HTTP Header or Query String).

  This procedure returns:

  - UserID: The derived UserID. If no UserID can be determined, return NULL.
  - Password: (Optional) The derived Password. If no Password can be determined, return NULL.

GetSSOUserID Procedure Information		
Parameter	**Input/Output**	**Type**
CookieNameList	IN	MVW_Authenticate_Custom.Varchar2List_T
CookieValueList	IN	MVW_Authenticate_Custom.Varchar2List_T

GetSSOUserID Procedure Information		
Parameter	**Input/Output**	**Type**
HeaderNameList	IN	MVW_Authenticate_Custom.Varchar2List_T
HeaderValueList	IN	MVW_Authenticate_Custom.Varchar2List_T
ParameterNameList	IN	MVW_Authenticate_Custom.Varchar2List_T
ParameterValueList	IN	MVW_Authenticate_Custom.Varchar2List_T
UserID	OUT	Varchar2
Password	OUT	Varchar2
RunProductCode	OUT	Varchar2

- **GetPackageVersion**: Sets a character string that corresponds to the version number of the current package body, along with each individual point release number.

GetPackageVersion Procedure Information		
Parameter	**Input/Output**	**Type**
VersionString	OUT	varchar2
VersionA	OUT	number
VersionB	OUT	number
VersionC	OUT	number
VersionD	OUT	number

- **PackageVersion**: Returns a character string that corresponds to the version number of the current package body.

PackageVersion Function Information	
Parameter	**Type**
return	varchar2

MVW_SERVLET_DTM_CUSTOM

Also known as the MarkView Installation Script, which is use to set up MarkView in individual schemas.

Authorized Function Information		
Parameter	**Input/Output**	**Type**
userID	IN	Varchar2
password	IN	Varchar2
realm	IN	Varchar2
DTMUserID	IN	Varchar2
RETURN		Boolean

PreDocumentDownload Procedure Information		
Parameter	**Input/Output**	**Type**
documentID	IN	Number
pageNumber	IN	Number
userID	IN	Varchar2
DTMUserID	IN	Varchar2
realm	IN	Varchar2
DTMWorkstationSerialNumber	OUT	Varchar2
DTMPrivilegeCode	OUT	Varchar2
DTMErrorPage	OUT	Varchar2
DTMAttachmentName	OUT	Varchar2
DTMErrorMessage	OUT	Varchar2
folderID	IN	Number Default null

PreCreateDocumentUpload Procedure Information		
Parameter	**Input/Output**	**Type**
filename	IN	Varchar2
fileNumber	IN	Number
fileSize	IN	Number
mimeType	IN	Varchar2
formTagName	IN	Varchar2
userID	IN	Varchar2
realm	IN	Varchar2
DTMDocumentTypeID	OUT	Number
DTMPrivilegeCode	OUT	Varchar2
DTMWorkstationSerialNumber	OUT	Varchar2
DTMDocumentDescription	OUT	Varchar2
DTMDocumentNumber	OUT	Number
DTMPageNumber	OUT	Number
DTMErrorPage	OUT	Varchar2
DTMSuccessPage	OUT	Varchar2
DTMErrorMessage	OUT	Varchar2
folderID	IN	Number Default null
FileDesc	IN	Varchar2 Default '**NONE**'
DocumentTypeID	IN	Number Default null

PreCreateDocumentUpload Procedure Information		
Parameter	**Input/Output**	**Type**
ObjectClassID	IN	varchar2 default null

PostCreateDocumentUpload Procedure Information		
Parameter	**Input/Output**	**Type**
documentID	IN	Number
fileNumber	IN	Number
filename	IN	Varchar2
fileSize	IN	Number
mimeType	IN	Varchar2
formTagName	IN	Varchar2
userID	IN	Varchar2
realm	IN	Varchar2
documentNumber	IN	Number
pageNumber	IN	Number
DTMErrorPage	OUT	Varchar2
DTMSuccessPage	OUT	Varchar2
DTMErrorMessage	OUT	Varchar2
folderID	IN	Number Default null
ObjectClassID	IN	varchar2 default null

PreDeleteDocument Procedure Information		
Parameter	**Input/Output**	**Type**
documentID	IN	Number
pageNumber	IN	Number
userID	IN	Varchar2
realm	IN	Varchar2
DTMPrivilegeCode	OUT	Varchar2
DTMWorkstationSerialNumber	OUT	Varchar2
DTMErrorPage	OUT	Varchar2
DTMSuccessPage	OUT	Varchar2

PostDeleteDocument Procedure Information		
Parameter	**Input/Output**	**Type**
documentID	IN	Number

PostDeleteDocument Procedure Information		
Parameter	**Input/Output**	**Type**
pageNumber	IN	Number
userID	IN	Varchar2
realm	IN	Varchar2
DTMErrorPage	OUT	Varchar2
DTMSuccessPage	OUT	Varchar2
DTMErrorMessage	OUT	Varchar2

SF_MAIL_GATEWAY_CUSTOM

Use this package for inbound email processing.

Do not perform database commits in this package.

This procedure is called once each time SF Mail Gateway connects to the database. Use it to provide values for the parameters that allow you to upload documents to the MarkView Document Transport Module.

- **Initialize**

Initialize Procedure Information		
Parameter	**Input/Output**	**Type**
SQLFlowUserID	OUT	Boolean
MarkViewUserID	OUT	Varchar2
MarkViewPassword	OUT	Varchar2
EnableSingleSignOn	OUT	Varchar2
SingleSignOnURL	OUT	Varchar2
RunProductCode	OUT	Boolean
ExpNotUsed		Exception

Use the ExpNotUsed exception to deactivate individual SF_MAIL_GATEWAY_CUSTOM procedures (for example, indicate that the procedure is not used). When raised from PreAcceptMessage, SQL*Flow Mail Gateway remembers not to call the PreAcceptMessage and ProcessAttachment procedures during the current session.

To activate these procedures, remove the statement that raises this exception from that procedure. By default, all of the following SF_MAIL_GATEWAY_CUSTOM procedures are inactivate (for example, raise this exception). When the exception is raised, all incoming mail is accepted and attachments are discarded.

- **PreAcceptMessage**: Called once for each inbound mail message received by the SQL*Flow Mail Gateway.

Parameter Descriptions for the PreAcceptMessage Procedure	
Parameter	**Description**
MailFrom	Email address of the originating user of the mail message (for example, jsmith@company.com).
MailFromName	Full name of the originating user of the mail message (for example, John Smith).
MailTo	Email address of the destination user of the mail message (for example, bjones@company.com).
MailCc	Email address of the Cc'd users of the mail messages(for example, bgates@company.com; bclinton@company.com).
MailReplyTo	Email address of the ReplyTo user of the mail messages(for example, jsmith@company.com).
MailSubject	Subject of the mail message.
MailBodyListCount	The number of records in MailBodyList.
MailBodyList	List of strings that together make up the body of the mail message. The first 2000 characters of the body go into list element 1, the second 2000 characters go into list element 2, and so on.
MailBodyFormat	Format of the mail message body. This is set to "HTML" or "Text".
AcceptMessage	Parameter that indicates if a message should be accepted. • TRUE: Accepts the message • FALSE: Rejects the message
CreateWorkItem	Parameter that indicates if a work item should be created for a message. • TRUE: Create a work item for the message • FALSE: Do not create a work item for the message Each Attachment list parameter contains a list of data in which each item corresponds to a single attachment. If the message has no attachments, these lists are empty.
AttachmentCount	The number of attachments.
AttachmentNameList	The original name of the attachment, for example, MyDocument.doc.
AttachmentSizeList	The size (in bytes) of the attachment.
AttachmentMIMETypeList	The primary content type, which is extracted from the "Content-Type:" attachment header and is only relevant in MIME formatted messages. For example, the following sample content-type attachment header record sets this parameter to "application". Content-Type: application/x-zip-compressed; name="smtp.zip"

Parameter Descriptions for the PreAcceptMessage Procedure	
Parameter	**Description**
AttachmentMIMESubTypeList	The sub content type which is extracted from the "Content-Type:" attachment header and is only relevant in MIME formatted messages. For example, the following sample content-type attachment header record sets the ContentSubType property to "x-zip-compressed". Content-Type: application/x-zip-compressed; name="smtp.zip"
AttachmentSplitCountList	The number of files into which this attachment is split. Some files types, for example, TIFF, can be split into multiple files. If the attachment is not split, this is set to 1.
AcceptAttachmentList	The parameter that indicates if the attachment should be accepted. • TRUE: Accepts the attachment • FALSE: Rejects the attachment
SplitAttachmentList	If the attachment is accepted, this parameter specifies if the attachment can be split. If the attachment is not accepted or cannot be split, this parameter is ignored. • TRUE: Allows the attachment to be split • FALSE: Does not allow the attachment to be split
WorkItemClassName	The name of the Work Item Class to use when SQL*Flow Mail Gateway creates the Work Item. By default, this parameter is set to the Work Item Class specified in the Preferences window of SQL*Flow Mail Gateway. The Work Item Class must meet the requirements of a SQL*Flow Mail Gateway Work Item Class. It must have MailTo, MailFrom, MailSubject and MailBody Work Item Properties defined. Use the Work Item Property parameters to specify Work Item Property values to use when the Work Item Class (as specified by the Work Item Class preference in SQL*Flow Mail Gateway) is created. One possible use for this might be to specify the Work Item Property Name and Value for the MarkView Document ID that was generated for the files specified by the attachment).
WorkItemPropertyCount	The number of Work Item Properties in the lists.
WorkItemPropertyNameList	A 1-based list of Work Item Property Names for use when creating the Work Item Instance, for example, WorkItemPropertyNameList(1) := 'MVDocumentID'; The specified Work Item Property Names must be valid for the Work Item Class preference in SQL*Flow Mail Gateway.
WorkItemPropertyValueList	A 1-based list of Work Item Property Names for use when creating the Work Item Instance , for example, WorkItemPropertyValueList(1) := '5465';

PreAcceptMessage Procedure Information		
Parameter	**Input/ Output**	**Type**
MailFrom	IN	Varchar2
MailFromName	IN	Varchar2
MailTo	IN	Varchar2
MailCc	IN	Varchar2
MailReplyTo	IN	Varchar2
MailSubject	IN	Varchar2
MailBodyListCount	IN	Number
MailBodyList	IN	SF_Mail_Gateway.MailBodyList_T
MailBodyFormat	IN	Varchar2
AcceptMessage	OUT	Boolean
CreateWorkItem	OUT	Boolean
AttachmentCount	IN	Number
AttachmentNameList	IN	SF_Mail_Gateway.FilenameList_T
AttachmentSizeList	IN	SF_Mail_Gateway.NumberList_T
AttachmentMIMETypeList	IN	SF_Mail_Gateway.MIMETypeList_T
AttachmentMIMESubTypeList	IN	SF_Mail_Gateway.MIMETypeList_T
AttachmentSplitCountList	IN	SF_Mail_Gateway.NumberList_T
AcceptAttachmentList	OUT	SF_Mail_Gateway.BooleanList_T
SplitAttachmentList	OUT	SF_Mail_Gateway.BooleanList_T
WorkItemClassName	IN OUT	Varchar2
WorkItemPropertyCount	IN OUT	Number
WorkItemPropertyNameList	IN OUT	SF_Client.WorkItemPropertyNameList_T
WorkItemPropertyValueList	IN OUT	SF_Client.WorkItemPropertyValueList_T
RunProductCode	OUT	Boolean

- **ProcessAttachment**: Called once for each accepted attachment to each inbound email message received by the SQL*Flow Mail Gateway.

Parameter Descriptions for the ProcessAttachment Procedure	
Parameter	**Description**
MailFrom	Email address of the originating user of the mail message (for example, jsmith@company.com).
MailFromName	Full name of the originating user of the mail message (for example, John Smith).

Parameter Descriptions for the ProcessAttachment Procedure	
Parameter	**Description**
MailTo	Email address of the destination user of the mail message (for example, bjones@company.com)
MailCc	Email address of the Cc'd users of the mail messages (for example, rpratt@company.com; cstupak@company.com)
MailReplyTo	Email address of the ReplyTo user of the mail messages (for example, jsmith@company.com)
MailSubject	Subject of the mail message.
MailBodyListCount	The number of records in MailBodyList.
MailBodyList	List of strings that make up the body of the mail message. The first 2000 characters of the body go into list element 1, the second 2000 characters of the body go into list element 2, and so on.
MailBodyFormat	Format of the mail message body. Set this to "HTML" or "Text".
AttachmentCount	The total number of attachments in the entire message.
AttachmentIndex	The number of the attachment within the message. The first attachment has an Index of 1, the second has an index of 2, and so forth.
AttachmentName	The original name of the attachment (for example, MyDocument.doc).
AttachmentSize	The size (in bytes) of the attachment.
AttachmentMIMEType	The primary content type, which is extracted from the "Content-Type:" attachment header and is only relevant in MIME formatted messages. For example, the following sample content-type attachment header record sets this parameter to "application". Content-Type: application/x-zip-compressed; name="smtp.zip"
AttachmentMIMESubType	The sub content type, which is extracted from the "Content-Type:" attachment header and is only relevant in MIME formatted messages. For example, the following sample content-type attachment header record sets the ContentSubType property to "x-zip-compressed". Content-Type: application/x-zip-compressed; name="smtp.zip"
AttachmentSplitCount	The number of files into which this attachment is split. Some files types, for example, TIFF, can be split into multiple files. If the attachment is not split, this is set to 1.
AttachmentSplitFilenameList	A list of files that resulted from accepting the attachment. If the attachment was split into multiple files, this list contains those filenames in the order in which they were extracted from the original file. If the attachment was not split, this list contains only one item, which has the same value as AttachmentName.
AttachmentSplitSizeList	The size (in bytes) of the split file.

Parameter Descriptions for the ProcessAttachment Procedure	
Parameter	**Description**
RejectThisAttachment	Setting that directs the system to reject the attachment for this message. If any of the ProcessAttachment calls for this message set RejectAllAttachments to TRUE, this value is ignored. • TRUE: Reject this attachment • FALSE: Accept this attachment
RejectAllAttachments	Setting that directs the system to reject all attachments for this message. • TRUE: Rejects all attachments • FALSE: Accepts all attachments
AttachmentDestFilenameList	For each AttachmentFileSplit, the developer must use this 1-based list to specify the complete filename of the corresponding destination file. Complete filename specification must be in and EIT compatible format(for example, ftp://user:pwd@imgserver/images/1/4/2/3/00001423.tif').
WorkItemProperties	Settings that specify the values to use when creating the Work Item Class (as specified by the Work Item Class preference in SQL*Flow Mail Gateway). These lists are set to the current value of the Work Item Properties as set by the PreAcceptMessage procedure or by other calls to this procedure for the same mail message. You can use this to specify the Work Item Property Name and Value for the MarkView Document ID that was generated for the files specified by the attachment.
WorkItemPropertyCount	The number of Work Item Properties in the lists.
WorkItemPropertyNameList	A 1-based list of Work Item Property Names for use when creating the Work Item Instance, for example, WorkItemPropertyNameList(1) := 'MVDocumentID'; Work Item Property Names must be valid for the Work Item Class preference in SQL*Flow Mail Gateway.
WorkItemPropertyValueList	A 1-based list of Work Item Property Values for use when creating the Work Item Instance, for example, WorkItemPropertyValueList(1) := '5465';

ProcessAttachment Procedure Information		
Parameter	**Input/Output**	**Type**
MailFrom	IN	Varchar2
MailFromName	IN	Varchar2
MailTo	IN	Varchar2
MailCc	IN	Varchar2
MailReplyTo	IN	Varchar2
MailSubject	IN	Varchar2

ProcessAttachment Procedure Information		
Parameter	**Input/Output**	**Type**
MailBodyListCount	IN	Number
MailBodyList	IN	SF_Mail_Gateway.MailBodyList_T
MailBodyFormat	IN	Varchar2
AttachmentCount	IN	Number
AttachmentIndex	IN	Number
AttachmentName	IN	Varchar2
AttachmentSize	IN	Number
AttachmentMIMEType	IN	Varchar2
AttachmentMIMESubType	IN	Varchar2
AttachmentSplitCount	IN	Number
AttachmentSplitFilenameList	IN	SF_Mail_Gateway.FilenameList_T
AttachmentSplitSizeList	IN	SF_Mail_Gateway.NumberList_T
RejectThisAttachment	OUT	Boolean
RejectAllAttachments	OUT	Boolean
AttachmentDestFilenameList	OUT	SF_Mail_Gateway.CompleteFilenameList_T
WorkItemPropertyCount	IN OUT	Number
WorkItemPropertyNameList	IN OUT	SF_Client.WorkItemPropertyNameList_T
WorkItemPropertyValueList	IN OUT	SF_Client.WorkItemPropertyValueList_T
RunProductCode	OUT	Boolean

Do not perform any database commits in this package.

Chapter 2

Modify MarkView with join points

This chapter gives you information about join points and advice.

Join points

Joint points are predetermined locations in the product [PL SQL] code where you can modify the behavior of the original program by calling other code, called *advice*. The join point table for each join point contains information that determines which advice modifies product behavior.

For example, the MV_TOOL_JOIN_POINT join point table contains the following columns.

Column Name	Data Type
JOIN_POINT_ID	NOT NULL NUMBER(8)
TOOL_ID	NOT NULL NUMBER(8)
DOCUMENT_TYPE_ID	NOT NULL NUMBER(8)
JOIN_POINT_WHEN	NOT NULL VARCHAR2(30)
JOIN_POINT_TYPE	NOT NULL VARCHAR2(10)
JOIN_POINT_CALL	NOT NULL VARCHAR2(200)
MODULE_NAME	NOT NULL VARCHAR2(100)

Join Point Columns

JOIN_POINT_ID

A unique identifier. Every Join Point Table has this column and a different sequence number. For example, in the MV_TOOL_JOIN_POINT join point table, the JOIN_POINT_ID comes from the sequence MV_TOOL_JOIN_POINT_SEQ.

TOOL_ID

Maps to the MV_TOOL table. Unique to this join point.

DOCUMENT_TYPE_ID

Maps to MV_DOCUMENT_TYPE table. Unique to this join point.

JOIN_POINT_WHEN

Specifies the action to perform. For example, use the tool to create or edit a Markup on the given document type. JOIN_POINT_WHEN is a common column, but does not appear in every join point. The valid values depend on the specific join point.

JOIN_POINT_TYPE

Determines the order for the advice. All join points have a JOIN_POINT_TYPE column, but the valid types vary depending on the case:

- PRODUCT (default)
- BEFORE: advice called before PRODUCT advice
- AROUND: advice called instead of PRODUCT advice
- AFTER: advice called after PRODUCT advice

By default, the table includes PRODUCT advice in one line, however you can add additional advice lines. In the MV_TOOL_JOIN_POINT example, although only one PRODUCT line will contain a combination of TOOL_ID, DOCUMENT_TYPE_ID and JOIN_POINT_WHEN values, there may be additional lines for other values of JOIN_POINT_TYPE (BEFORE, AROUND, or AFTER). Each additional line has a separate JOINT_POINT_ID.

JOIN_POINT_CALL

The procedure name called when there is a match for the other items in this table.

MODULE_NAME

Always "Custom".

Advice

Advice is the PL/SQL code that executes at join points to alter the behavior of the product code. Advice can do the following:

- Change the values of input parameters to run the base code with modified variables
- Execute instead of the base code
- Perform tasks in addition to those performed by the base code

You can add advice to join points two ways:

- Use script inserts in Join Point tables
- Use the import module tool for Join Point tables

Advice must be in PL/SQL, in the form of procedures or functions. Advice can run **BEFORE** PRODUCT advice, **AROUND** (instead of) PRODUCT advice, or **AFTER** PRODUCT advice code.

Advice	Description
Tool Advice	
Pre-Create	Called when the user selects tool from the MarkView Viewer toolbar or clicks to place a tool on document image.
Pre-View	Called when the user selects a tool that is already on the document image.
Pre-Edit	Called when the user attempts to save changes to a tool that has been saved.
Pre-Delete	Called when a user attempts to delete a tool that has been saved.
LOV Advice	
On-Populate	Called when determining the values to show in an LOV.

Advice	Description
On-Control-Validate	Called when the user attempts to change focus to a different field.
On-Form-Validate	Called when user attempts to close a form. (Action Accept)
Post-Modify	Called after the tool is saved successfully.
Virtual Work Item Property Advice	
On-Populate	Called when system virtual work item property is accessed. (Invoice Number, Vendor Name)
Rule Advice	
On-Execute	Called when a rule is triggered (can occur before or instead of existing rule).
Security Advice	
Document Access	Called when the viewer calls the DTM.
SSI Field Configuration Advice	
Display Field	Called to control aspects of the display of a header or line field.
Validate Field	Called to validate the data in a field when the user tries to save the header or line.

Tool join points

Markup join points

Call Type: Post Modify Markup Object

Advice Type: BEFORE

Parameter	Mode	Data Type
DocumentData	IN OUT	MV_MARKUP_ADVICE.DocumentData_T
MarkupObjectData	IN OUT	MV_MARKUP_ADVICE.MarkupObjectData_T
UserData	IN OUT	MV_MARKUP_ADVICE.UserData_T
IsAuthorized	IN OUT	Boolean
ErrorMessage	IN OUT	Varchar2

Call Type: Post Modify Markup Object

Advice Type: AFTER

Parameter	Mode	Data Type
DocumentData	IN	MV_MARKUP_ADVICE.DocumentData_T
MarkupObjectData	IN	MV_MARKUP_ADVICE.MarkupObjectData_T
UserData	IN	MV_MARKUP_ADVICE.UserData_T

Parameter	Mode	Data Type
IsAuthorized	IN OUT	Boolean
ErrorMessage	IN OUT	Varchar2

Call Type: Post Modify Markup Object

Advice Type: AROUND, PRODUCT

Parameter	Mode	Data Type
DocumentData	IN	MV_MARKUP_ADVICE.DocumentData_T
MarkupObjectData	IN OUT	MV_MARKUP_ADVICE.MarkupObjectData_T
UserData	IN	MV_MARKUP_ADVICE.UserData_T
IsAuthorized	IN OUT	Boolean
ErrorMessage	OUT	Varchar2

Call Types: Pre Create, Pre View, Pre Edit, Pre Delete

Advice Type: BEFORE

Parameter	Mode	Data Type
DocumentData	IN OUT	MV_MARKUP_ADVICE.DocumentData_T
FormData	IN OUT	MV_MARKUP_ADVICE.FormData_T
UserData	IN OUT	MV_MARKUP_ADVICE.UserData_T
IsAuthorized	OUT	Boolean
ErrorMessage	OUT	Varchar2

Call Types: Pre Create, Pre View, Pre Edit, Pre Delete

Advice Type: AFTER

Parameter	Mode	Data Type
DocumentData	IN	MV_MARKUP_ADVICE.DocumentData_T
FormData	IN	MV_MARKUP_ADVICE.FormData_T
UserData	IN	MV_MARKUP_ADVICE.UserData_T
IsAuthorized	IN OUT	Boolean
ErrorMessage	IN OUT	Varchar2

Call Types: Pre Create, Pre View, Pre Edit, Pre Delete

Advice Type: PRODUCT

Parameter	Mode	Data Type
DocumentData	IN	MV_MARKUP_ADVICE.DocumentData_T
FormData	IN OUT	MV_MARKUP_ADVICE.FormData_T

Parameter	Mode	Data Type
UserData	IN	MV_MARKUP_ADVICE.UserData_T
IsAuthorized	IN OUT	Boolean
ErrorMessage	OUT	Varchar2

Call Types: Pre Create, Pre View, Pre Edit, Pre Delete

Advice Type: AROUND

Parameter	Mode	Data Type
DocumentData	IN	MV_MARKUP_ADVICE.DocumentData_T
FormData	IN OUT	MV_MARKUP_ADVICE.FormData_T
UserData	IN	MV_MARKUP_ADVICE.UserData_T
IsAuthorized	IN OUT	Boolean
ErrorMessage	OUT	Varchar2

Call Type: Validate Control

Advice Type: BEFORE

Parameter	Mode	Data Type
DocumentData	IN OUT	MV_MARKUP_ADVICE.DocumentData_T
FormData	IN OUT	MV_MARKUP_ADVICE.FormData_T
CurrentControlData	IN OUT	MV_MARKUP_ADVICE.CurrentControlData_T
UserData	IN OUT	MV_MARKUP_ADVICE.UserData_T
IsValid	OUT	Boolean
ErrorMessage	OUT	Varchar2

Call Type: Validate Control

Advice Type: AFTER

Parameter	Mode	Data Type
DocumentData	IN	MV_MARKUP_ADVICE.DocumentData_T
FormData	IN	MV_MARKUP_ADVICE.FormData_T
CurrentControlData	IN	MV_MARKUP_ADVICE.CurrentControlData_T
UserData	IN	MV_MARKUP_ADVICE.UserData_T
IsValid	IN OUT	Boolean
ErrorMessage	IN OUT	Varchar2

Call Type: Validate Form

Advice Type: BEFORE

Parameter	Mode	Data Type
DocumentData	IN OUT	MV_MARKUP_ADVICE.DocumentData_T
FormData	IN OUT	MV_MARKUP_ADVICE.FormData_T
UserData	IN OUT	MV_MARKUP_ADVICE.UserData_T
IsValid	OUT	Boolean
ErrorMessage	OUT	Varchar2

Call Type: Validate Form

Advice Type: AFTER

Parameter	Mode	Data Type
DocumentData	IN	MV_MARKUP_ADVICE.DocumentData_T
FormData	IN	MV_MARKUP_ADVICE.FormData_T
UserData	IN	MV_MARKUP_ADVICE.UserData_T
IsValid	IN OUT	Boolean
ErrorMessage	IN OUT	Varchar2

Call Type: Validate Form

Advice Type: PRODUCT

Parameter	Mode	Data Type
DocumentData	IN	MV_MARKUP_ADVICE.DocumentData_T
FormData	IN OUT	MV_MARKUP_ADVICE.FormData_T
UserData	IN	MV_MARKUP_ADVICE.UserData_T
IsValid	IN OUT	Boolean
ErrorMessage	OUT	Varchar2

Call Type: Validate Form

Advice Type: AROUND

Parameter	Mode	Data Type
DocumentData	IN	MV_MARKUP_ADVICE.DocumentData_T
FormData	IN OUT	MV_MARKUP_ADVICE.FormData_T
UserData	IN	MV_MARKUP_ADVICE.UserData_T
IsValid	IN OUT	Boolean
ErrorMessage	OUT	Varchar2

List of Values join points

Call Type—On Populate

Advice Type—BEFORE, PRODUCT, AROUND, AFTER

Parameter	Mode	Data Type
LOVData	IN OUT	MV_LOV_ADVICE.LOVData_T

Virtual Work Item Property join points

Call Type—Function

Advice Type—AROUND, PRODUCT

Parameter	Mode	Data Type
WorkItemInstanceID	IN	Number
RETURN	Return	Varchar2

Join Points working with Virtual Work Item Property values must execute the following Function:

```
function SF_RULE_PUBLIC.WorkItemProp (PropName IN Varchar2)
return Varchar2
```

Rule join points

Rule Call Types

Advice Type—BEFORE

Parameter	Mode	Data Type
Result	OUT	Number

Advice Type—AROUND

Parameter	Mode	Data Type
Result	OUT	Number

SSI join points

SSI has two types of join points:

Display Field
Controls aspects of the display of a header or line field.

Validate Field
Validates the data in a field when the user tries to save the header or line.

SSI Field Display join points

Field Display Type Header

Advice Type—AROUND, AFTER

Parameter	Mode	Data Type
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
InvoiceHeaderData	IN	MVOA_SSI_Invoices.InvoiceHeaderRecord_T
DefaultValue	IN OUT	Varchar2
ShowField	IN OUT	Boolean
Required	IN OUT	Boolean
ReadOnly	IN OUT	Boolean

Field Display type header LOV

Advice Type—AROUND, AFTER

Parameter	Mode	Data Type
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
InvoiceHeaderData	IN	MVOA_SSI_Invoices.InvoiceHeaderRecord_T
SupplierSiteDefaultValue	IN	Varchar2
ValueList	IN	mvoa_ssi_ui_invoice_controls.ValueList_T
DescriptionList	IN	mvoa_ssi_ui_invoice_controls.DescriptionList_T
ListCount	IN	number
DefaultValue	IN	Varchar2
IndexList	OUT	mvoa_ssi_ui_controls.InvoiceList_T
IndexListCount	IN OUT	number
DefaltValueIndex	IN OUT	number
ShowField	IN OUT	Boolean
Required	IN OUT	Boolean
ReadOnly	IN OUT	Boolean

Field Display Type Line

Advice Type—AROUND, AFTER

Parameter	Mode	Data Type
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
InvoiceHeaderData	IN	MVOA_SSI_Invoices.InvoiceHeaderRecord_T
DefaultValue	IN OUT	Varchar2
ShowField	IN OUT	Boolean
Required	IN OUT	Boolean
ReadOnly	IN OUT	Boolean

Field Display Type Line LOV

Advice Type—AROUND, AFTER

Parameter	Mode	Data Type
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util. RequestData_T
InvoiceHeaderData	IN	MVOA_SSI_Invoices.InvoiceHeaderRecord_T
ValueList	IN	mvoa_ssi_ui_invoice_controls.ValueList_T
DescriptionList	IN	mvoa_ssi_ui_invoice_controls.DescriptionList_T
ListCount	IN	number
DefaultValue	IN	Varchar2
IndexList	OUT	mvoa_ssi_ui_controls.InvoiceList_T
IndexListCount	IN OUT	number
DefaltValueIndex	IN OUT	number
ShowField	IN OUT	Boolean
Required	IN OUT	Boolean
ReadOnly	IN OUT	Boolean

SSI Field Validation Join Points

Field Validation Type Header

Advice Type—AROUND, AFTER

Parameter	Mode	Data Type
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
InvoiceHeaderData	IN	MVOA_SSI_Invoices.InvoiceHeaderRecord_T
InvoiceDistributionData	IN	MVOA_SSI_Invoices.InvoiceDistributionRecord_T

Parameter	Mode	Data Type
Value	IN	Varchar2
IsValid	IN OUT	Boolean
ErrorMessage	IN OUT	Varchar2

Field Validation Type Line

Advice Type—AROUND, AFTER

Parameter	Mode	Data Type
UserID	IN	mv_user_profile.user_id%TYPE
RequestData	IN	MVOA_SSI_Util.RequestData_T
InvoiceHeaderData	IN	MVOA_SSI_Invoices.InvoiceHeaderRecord_T
Value	IN	Varchar2
IsValid	IN OUT	Boolean
ErrorMessage	IN OUT	Varchar2

Chapter 3

Set up advanced security

Set up Single Sign-On

A third-party authentication source authenticates users or verifies that a user is already logged in, and then presents MarkView with the relevant user credentials. Typically when MarkView is integrated with a single sign-on (SSO) provider, user credentials are passed into MarkView either by an HTTP Cookie/Header or by the Security Assertion Markup Language standard (SAML). SSO is widely used for end-user authentication but not for Capture and Output components.

MarkView supports the following SSO configurations:

- Standard Cookie/Header SSO Configuration

  MarkView does not verify user credentials but uses the credentials provided by the SSO system. The SSO system (agent) controls physical access to MarkView for authenticated users only and blocks unauthorized access.
- Custom Cookie/Header SSO Configuration

  MarkView passes all HTTP request headers, cookies, and parameters to the custom PL/SQL code for the authentication.
- SAML authentication via WebSSO profile

  Kofax recommends that you use SAML to enable SSO in MarkView.

MarkView 9.1.0 and higher does not require WAR or EAR file modification for the supported SSO configurations.

Enable SSO

Use the Standard Cookie/Header SSO configuration if your SSO implementation is based on passing the user ID in the cookie or header.

Use the Custom Cookie/Header SSO configuration if your SSO implementation is based on passing any custom information (not the user ID) in the cookie or header.

For Custom Cookie/Header SSO configuration, all cookies and headers are passed into the MVW_Authenticate_Custom.GetSSOUserID procedure. The current authenticated user's MarkView user ID is in the list of headers and cookies that are passed into the procedure.

In some environments, a user ID is not available but a session ID is passed. The session ID can be used to determine the user ID. Typically, a web service or database API is provided to retrieve a user ID from a session ID. The user ID and password (if available) is then passed into the

MVW_Authenticate_Custom.Authenticate procedure. This procedure should be modified to validate the user credentials and not only return the default value of false.

Use the SAML SSO configuration if your SSO implementation is based on passing user credentials to MarkView by SAML.

Standard Cookie/Header SSO configuration

Use the Standard Cookie/Header SSO configuration if your SSO implementation is based on passing the user ID in the cookie or header.

1. Obtain the name of a cookie or header containing a user name (provided by the configured SSO agent) and the SSO login page URL.
2. Navigate to **Administration** > **MarkView Admin** and click **Preferences**.
3. Configure one of the following preferences:
   - AUTH_SSO_USERNAME_COOKIE_NAME: If your SSO implementation is based on passing the user ID in the cookie
   - AUTH_SSO_USERNAME_HEADER_FIELD: If your SSO implementation is based on passing the user ID in the header
4. Set the AUTH_LOGIN_HTTP_URL preference to SSO login URL.
5. Configure the following preferences to enable the SSO session termination, so that a user is logged out of MarkView before redirecting to SSO logout URL:
   a. Set the AUTH_LOGOUT_LANDING_URL preference to SSO logout URL.
   b. Set the AUTH_LOGOUT_HTTP_URL preference to the default value: logout.do

   If you set the AUTH_LOGOUT_HTTP_URL preference to SSO logout URL, MarkView does not perform a user logout before redirecting to SSO logout URL.
6. Set MV_ENABLE_SINGLE_SIGN_ON to **TRUE**.
7. Verify that the WEBCLNT_SEC_AUTH_TYPE preference specifies your authentication method and is not set to **CUSTOM**.
8. Restart MarkView applications or MarkView Server.
9. Go to MarkView home and verify that it redirects you to the SSO login page.
10. Authenticate against the SSO system and go to MarkView.
11. MarkView must authenticate the user based on the user name from the cookie or header provided by the SSO agent, or redirect to the login page if the authentication was unsuccessful.

The SSO system should allow all MarkView servers to have access to the `/usermgmt/rpc/` URL address for inter-apps communication. Refer to your specific SSO provider documentation for configuration instructions.

Custom Cookie/Header SSO configuration

If required, contact Kofax Professional Services for assistance in configuring this SSO implementation.

Use the Custom Cookie/Header SSO configuration if your SSO implementation is based on passing any custom information (not the user ID) in the cookie or header.

1. Obtain the name of a cookie or header containing a user name (provided by the configured SSO agent) and the SSO login page URL.
2. Navigate to **Administration** > **MarkView Admin** and click **Preferences**.
3. Configure one of the following preferences:
   - AUTH_SSO_USERNAME_COOKIE_NAME: If your SSO implementation is based on passing the user ID in the cookie
   - AUTH_SSO_USERNAME_HEADER_FIELD: If your SSO implementation is based on passing the user ID in the header
4. Set the AUTH_LOGIN_HTTP_URL preference to SSO login URL.
5. Configure the following preferences to enable the SSO session termination, so that a user is logged out of MarkView before redirecting to SSO logout URL:
   a. Set the AUTH_LOGOUT_LANDING_URL preference to SSO logout URL.
   b. Set the AUTH_LOGOUT_HTTP_URL preference to the default value: logout.do

   If you set the AUTH_LOGOUT_HTTP_URL preference to SSO logout URL, MarkView does not perform a user logout before redirecting to SSO logout URL.
6. Set MV_ENABLE_SINGLE_SIGN_ON to **TRUE**.
7. Set WEBCLNT_SEC_AUTH_TYPE to **CUSTOM** on the system level.
8. Verify that MVW_Authenticate_Custom.GetSSOUserID returns a correct user name based on the cookies, headers, and parameters.
9. Verify that MVW_Authenticate_Custom.Authenticate returns `true` for the data returned by the MVW_Authenticate_Custom.GetSSOUserID function.
10. Restart MarkView applications or MarkView Server.
11. Open MarkView and verify that it redirects you to the SSO login page.
12. Authenticate against the SSO system and go to MarkView.
13. MarkView must authenticate the user based on the user name from the cookie or header provided by the SSO agent or redirect to the login page if the authentication was unsuccessful.

The SSO system should allow all MarkView servers to have access to the `/usermgmt/rpc/` URL address for inter-apps communication. Refer to your specific SSO provider documentation for configuration instructions.

SAML SSO configuration

Use the SAML SSO configuration if your SSO implementation is based on passing user credentials to MarkView by the SAML standard.

Before you proceed to configuring SAML SSO, verify that:

- The Identity Provider (IdP) administrator configured SAML Identity Provider (SAML IdP).
- A cryptographic key pair has been generated. Contact to your IdP administrator for assistance.
- You configured the Java keystore and created a new cryptographic key pair consisting of a private key and a public key.

  This key pair must be unique and it must be used only in MarkView.

- You added the key pair into the keystore. You must know its entry name and its password in the keystore.
- The IdP administrator is informed that the user ID is sent as an attribute of the SAML authentication response. The user ID must exactly match the user ID in MarkView.

  You must know the attribute name. The data type is `SAML2String` without encoding.
- Time synchronization via NTP server is configured for both MarkView and IdP servers. MarkView and IdP server time zones may vary.

1. Copy the Java keystore and IdP metadata file to `<markview_installation_directory>/<saml>`.
2. Navigate to **Administration** > **MarkView Admin** and click **Preferences**.
3. For the AUTH_SSO_SAML_KEYSTORE_PATH preference, specify the path to the Java keystore in the following format:

   `file:/projects/mvhome/markview/saml/saml.jks`

   Verify that you start the path with "file:" for both Unix and Windows systems.
4. For the AUTH_SSO_SAML_KEYSTORE_PASSWD preference, specify the password for the Java keystore.
5. For the AUTH_SSO_SAML_CERT_NAME preference, specify the name of the key pair in the Java keystore.

   The default value is saml.
6. For the AUTH_SSO_SAML_CERT_PASSWD preference, specify the password to the key pair entry in the Java keystore.
7. For the AUTH_SSO_SAML_USER_ATTRIBUTE_NAME preference, specify the SAML principal attribute name containing a user ID that should be used by MarkView to identify a user in the internal database.

   The default value is:

   urn:oid:0.9.2342.19200300.100.1.1
8. For the AUTH_SSO_SAML_IDP_METADATA_PATH preference, specify the path to the IdP metadata file in the following format:

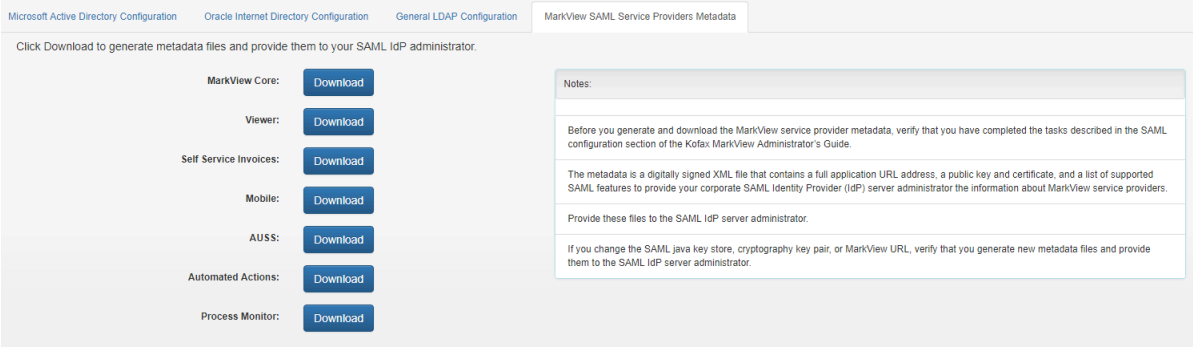
   `file:/projects/mvhome/markview/saml/idp-metadata.xml`

   Verify that you start the path with "file:" for both Unix and Windows systems.
9. Restart MarkView Server to enable the configured preferences.

   If the SAML preferences have incorrect values, the MarkView applications might fail to start. Reset the preferences using an SQL script as described in Reset SAML preferences.

10. If all MarkView applications start without errors, export the MarkView metadata:
    a. Navigate to **Administration** > **Authentication Configuration**.
    b. On the **MarkView SAML Service Providers Metadata** tab, click **Download** for all items in the list.
    c. Save all generated XML files and provide them to the IdP administrators.

    An XML file contains a full application URL address, a public key and certificate, and a list of supported SAML features to provide your corporate SAML IdP server administrator the information about MarkView service providers.



If you change the SAML Java key store, cryptography key pair, or MarkView URL, verify that you generate new metadata files and provide them to the SAML IdP server administrator.

11. Enable SAML authentication in MarkView:
    a. Set the AUTH_SSO_SAML_ENABLE_SUPPORT preference to **TRUE**.
    b. Set the MV_ENABLE_SINGLE_SIGN_ON preference to **TRUE**.
    c. Restart MarkView Server.

Other considerations

Redirect to the SSO server login page

The initial attempt to access MarkView might fail because a user has not previously logged in to the SSO server that provides MarkView with HTTP cookie or header values. If the AUTH_LOGIN_HTTP_URL preference has the default value, the user will be prompted to sign in as a regular MarkView user. You must set the AUTH_LOGIN_HTTP_URL preference to SSO login URL to redirect the user to a third-party SSO server login page where the proper authentication can be handled.

Single Logout

Use the Single Logout (SLO) SAML feature to log out from all participating applications in a created session.

Set the AUTH_SSO_SAML_ENABLE_LOGOUT_SUPPORT preference to TRUE to apply Single Logout and terminate the SSO session.

If a user logs out of a non-MarkView SSO application, the current MarkView session does not end.

If you set the AUTH_SSO_SAML_ENABLE_LOGOUT_SUPPORT preference to FALSE, the logout action terminates only the local MarkView session and does not affect either a session at IdP, or other application sessions where a user is logged in using single sign-on.

After you set the AUTH_SSO_SAML_ENABLE_LOGOUT_SUPPORT preference to FALSE, configure one of the following preferences:

- AUTH_LOGOUT_LANDING_URL: Sets the URL address to redirect to after logout.
- AUTH_HIDE_LOGOUT_LINK: Hides the logout link for all MarkView pages. The settings do not affect the mobile application.

For more information about MarkView authentication preferences, see the "MarkView Preferences" chapter of *Kofax MarkView Administrator's Guide, Volume 1*.

Set web browser access to SQL procedures

MarkView lets you restrict access to SQL procedures called from a web browser using entries that appear in the MVT_WEB_ACCESS_LIST table. Each line in the table contains a procedure name, group name, access setting, and the status of the line.

If the procedure called by a user matches a procedure name in the table, and the user ID matches a user ID in the table, and the user belongs to a group in the table, MarkView allows or disallows access based on the ACCESS_ALLOWED_YN setting for that line. If set to Y, MarkView grants access to the procedure; if N, MarkView denies access. When a user is denied access, MarkView adds an entry into mvplsql.log file and generates a message similar to the following:

```
You attempted to access an unauthorized web page. Please contact your MarkView
 Administrator.
```

The following shows the default MVT_WEB_ACCESS_LIST table, which the administrator can change. (During MarkView upgrades, this table is subject to change.)

WEB_ACCESS_LIST_ID	PROCEDURE_NAME	GROUP_NAME	USER_ID	ACCESS_ALLOWED_YN	ENABLED_YN	MODULE_NAME
2	MVAP_%	INTERACTIVE QUERIES	%	Y	Y	core-apps
3	SF_ADMIN_%	%MODULE_ADMINISTRATOR	%	Y	Y	core-apps
4	MV_ADMIN_%	%MODULE_ADMINISTRATOR	%	Y	Y	core-apps
5	MVRM_%	%	%	Y	Y	core-apps
6	MVSAP_%	%MODULE_ADMINISTRATOR	%	Y	Y	core-apps
7	%	%	%	Y	N	core-apps
8	MVW_%	%	%	Y	Y	core-apps
9	MVERP_%	%MODULE_ADMINISTRATOR	%	Y	Y	core-apps
10	MVOA_SSI_%	%MODULE_ADMINISTRATOR	%	Y	Y	core-apps
11	MVAP_%	%MODULE_ADMINISTRATOR	%	Y	Y	core-apps
12	%	%MODULE_ADMINISTRATOR	%	Y	N	core-apps
13	MVT_%	%	%	Y	Y	core-apps
14	MVAP_%	MARKVIEW WEB ADMINISTRATOR	%	Y	Y	core-apps
18	MVX_%	%	%	Y	Y	core-apps
19	MVC_%	%	%	Y	Y	core-apps

Modify MVT_WEB_ACCESS_LIST table entries directly by using SQL scripts, as in the following example.

```
insert into mvt_web_access_list (
  web_access_list_id, procedure_name, group_name,
  user_id, access_allowed_yn, module_name)
values (MVT_WEB_ACCESS_LIST_ID_SEQ.nextval,'MVX_%','%','%','Y','core-apps');

insert into mvt_web_access_list (
  web_access_list_id, procedure_name, group_name,
  user_id, access_allowed_yn, module_name)
values (MVT_WEB_ACCESS_LIST_ID_SEQ.nextval,'MVC_%','%','%','Y','core-apps');
```

Chapter 4

Set up MarkView Viewer

This chapter describes the MarkView Viewer administration, which you control through MarkView preferences.

Configure the default MarkView Viewer layout

To enable a user to configure the MarkView Viewer layout, add them to the VIEWER ADMINISTRATOR user group in **Administration** > **MarkView Admin** > **User Groups**. The Settings Menu option becomes available in MarkView Viewer to the members of the VIEWER ADMINISTRATOR group.

Use layout settings to enable a user to load, save, delete, and configure the default layout on the user, group, or system levels.

Configure authentication

After installing MarkView, configure the authentication method.

When the employee accesses MarkView Viewer, MarkView prompts the employee for a user name and password. If the system is not properly configured, the employee might be prompted for a user name and password multiple times.

Configuring the authentication method is a 3-step process, which you complete in the following order:

1. Configuring the Realm
2. Using Apache as a Proxy Server
3. Configuring the viewer to access the Apache Server

Configure the realm parameter

The realm is the web space against which MarkView authenticates users.

To configure the realm for MarkView, set the MVW_SECURITY_REALM preference to the same value as the mv.realm parameter in the web.xml file (Application_Server_HOME/viewer/viewer/WEB-INF/web.xml).

Restart the viewer instance of application server to implement the changes.

Use Apache as a proxy server

Configure MarkView to use Apache as a proxy server.

For information about configuring Apache, see the Apache documentation.

If you configure Oracle HTTP Server to proxy requests to application server running the viewer, enable SSL in the Oracle HTTP Server, not in the application server. If the application server instance is running SSL, you receive "Internal Error: 500" when trying to retrieve session information.

1. On your Apache server, locate the httpd.conf file and open it for editing.
2. Add the following directives to the Proxy Server directives section of the file:

   ```
   <IfModule mod_proxy.c>
   ProxyRequests On
   ProxyPass /viewer http://host:port/viewer
   ProxyPassReverse /viewer http://host:port/viewer
   ProxyPass /docserver http://host:port/docserver
   ProxyPassReverse /docserver http://host:port/docserver
   </IfModule>
   ```

   Where <host> is the host name of the system where you installed MarkView Viewer, and <port> is the HTTP port for the MarkView application server instance.
3. Save and close the file.

Configure MarkView Viewer to access the Apache server

Change the URLs specified in MarkView preferences to reference the proxy server.

1. Log in to MarkView and navigate to **Administration** > **MarkView Admin**.
2. Select the **Preferences** tab.
3. Search for the preferences to update:
   - **VIEWER_HTTP_BASE_URL**
   - **DOCSERVER_HTTP_BASE_URL**
4. Click the **Details** button next to the preference.
5. Select the **System Preferences** tab.
6. Click **Details**.
7. Replace the host name and port number in the URL with the host name and port number of the proxy server that you set up.

   For example, if the value of the VIEWER_BASE_URL preference is:

   http://myhost:myport/viewer

   update it to:

   http://proxyhost:proxyport/viewer
8. Click **Save**.
9. Repeat steps 3-8 for both preferences.

MarkView logging

MarkView uses the open source logging Log4J to log error and other debug activity during user sessions.

Use the VIEWER_LOGGING_LEVEL preference to control the MarkView Viewer logging level.

When you set a logging level preference to a valid logging level, the system performs logging for all classes at the specified level or a lower level.

Logging levels

Log4J has logging levels of off, fatal, error, warn, info, debug, and trace. The table compares the MarkView logging levels to equivalent Log4J logging level.

Log4J	MarkView
Off	OFF
Fatal	FATAL
Errors	ERROR
Warning	WARN
Information	INFO
Debug	DEBUG
Trace	ALL

To log all ERROR, WARN, and INFO messages, set the preference to INFO.

Log files

The log4j.properties file identifies the log files: log/viewer.log and log/docserver.log by default. The log4j.properties file specifies the format of messages written to the log files. To change the message format, see the javadoc org.apache.log4j.PatternLayout. See the Log4J Project Web site for additional information:

http://logging.apache.org/log4j/

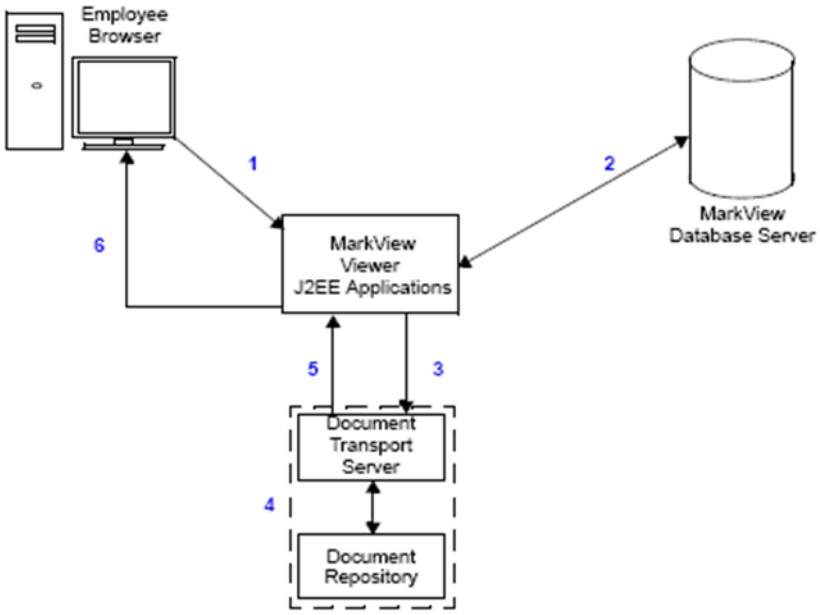
The viewer.log file identifies the MarkView Viewer version number during initialization when you set the VIEWER_LOGGING_LEVEL preference to at least INFO. For example:

```
2011-09-03 11:52:42,571 [OC4J Launcher] INFO  com.markview.common.Version:71 - Module
 version is: 7.1.0.32434
2011-09-03 11:52:42,571 [OC4J Launcher] INFO  com.markview.common.Version:71 - Module
 build date is: Thu Jun 03 11:44:03 EDT 2011
```

Have this information available if Kofax Technical Support asks for the version number during a support session.

Data flow

The data flow process involves exchanges between several components to display a document in MarkView Viewer. Numbers following the figure correspond to those in the figure.



MarkView Viewer Data Flow

1. An employee clicks an item in the Working Folder or Console to view.
2. The viewer retrieves ERP data from the database, which includes the URL for the image.
3. The viewer calls the DTM to retrieve the image file from the Document Server. The DTM authenticates the user who initiated the request.
4. The DTM retrieves the image. The combination of volume path and workstation serial number determines the method used. For the best results, co-locate DTM and Document Server on the same host.
5. The DTM returns the image file to the viewer.
6. The viewer renders the image, combines it with the ERP data from Step 2, and presents it within the viewer.

Chapter 5

Configure tools

Tools are Annotations and Actions that MarkView users place on documents. If you click the Tools button in MarkView Administration, a list of all the tools currently available on your system appears.

The following example shows a partial list of tools that may be available.

	Tool ID	Tool Name	Description
Details	505	Approved	Approve
Details	539	Archive	Move this to the Archive Queue
Details	514	Attach Follow-Up Document	Update property values required to auto attach document.
Details	529	Blackout	Blacks out the specified region of the image
Details	534	Blue Sticky Note	Places a blue sticky note at the specified location
Details	502	Cancelled Invoice	Cancelled Invoice
Details	526	Comment Requested	Comment Requested
Details	547	Comments	Adds a placeholder for the Comments
Details	513	Confidential	Mark this as confidential
Details	544	Copy Document	Creates a duplicate of the current document
Details	542	Delete Page	Delete Page

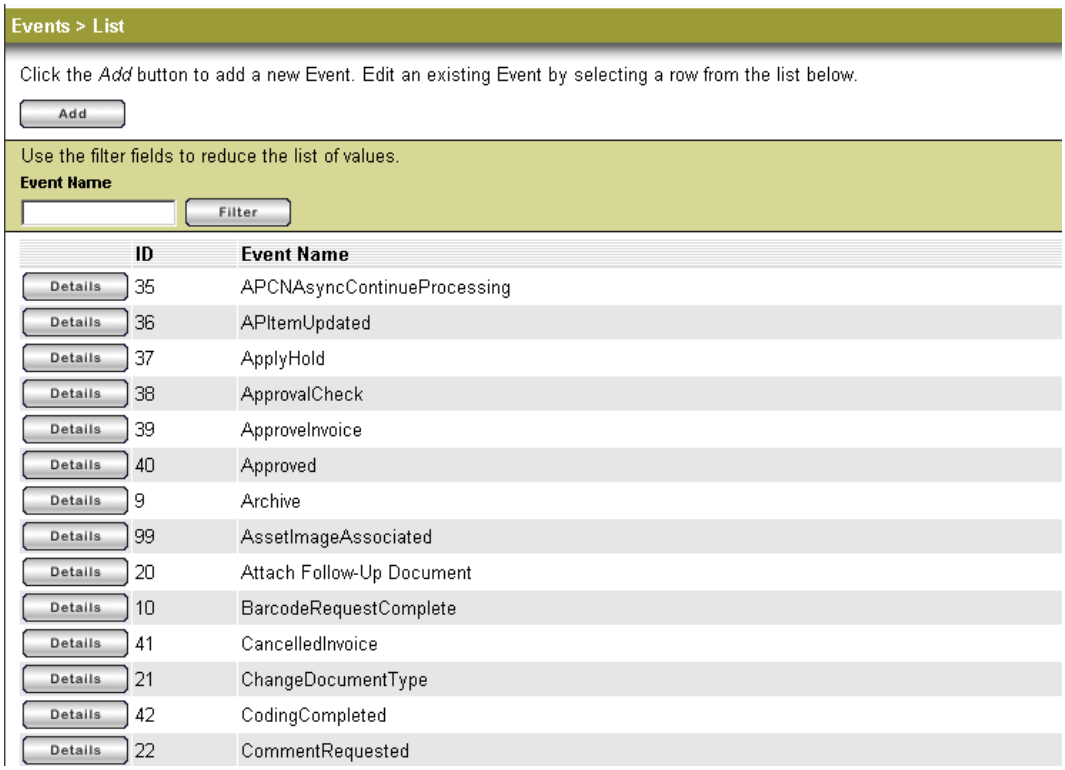
If you do not assign a Tool Queue Authorization (Tool Queue Auth) to a tool, the tool is available in all queues. If you assign a Tool Queue Auth, the tool becomes enabled in only those queues specified in the authorization window. The tool is disabled in other queues.

Enable tools for queues

1. Log in to MarkView and navigate to **Administration** > **Module Admin**.
2. Select **Tool Queue Auth**.
   The MVT Tool Queue Admin page appears.
3. Click **Insert**, and scroll to the bottom of the page.
4. In the **Tool Name** list, select the **tool** to activate.
5. In the **Queue Name** list, select the **queue** for which the tool will be active.
6. Click **Insert**.

Associate tools with events

Events trigger workflow processing. If you click the Event Types button in MarkView Process Administration, a list of all the events currently available on your system appears. For example, the following figure shows a partial list of events available on a sample system.



To facilitate workflow processing, define an event that receives an alert when an employee places a tool on a document. If you do not associate an event with a tool, placing the tool does not trigger workflow processing. You can associate multiple tools with the same event.

1. Log in to MarkView and navigate to **Administration** > **Module Admin**. .
2. Select **Tool Event Auth**.
   The MVT Tool Event Admin page appears.
3. Click **Insert**, then scroll to the bottom of the page.
4. In the **Tool Name** list, select a tool.
5. In the **Event Name** list, select the event to alert when the tool is placed on a document.
6. Click **Insert**.
   For example, to associate the Rescan Rejected tool with the Rescan Rejected event, set the menus as shown in the following figure.

Set tool security options

MarkView provides security options that let you control which annotations a user group can access for a document type. The security options also let you specify the level of access to grant.

For example, you can hide an annotation from one user group yet give another group the ability to edit the annotation. MarkView provides security for the following annotations in the viewer.

Blackout	Red Arrow
BlueSticky Note	Red Line
Comments	Red Text
Delete Page	Subject
Green Ellipse	Submitting Username
Green Text	Total Pages
New Document	Whiteout
Page Number	Yellow Highlight
Recipient Company	Yellow Sticky Note
Recipient Name	Custom tools

1. Log in to MarkView and navigate to **Administration** > **MarkView Admin**.
2. Select the **User Groups** tab.
3. Locate the user group to modify, and click **Details** next to the group name.
4. Select the **Tool Privilege Auths** tab.
5. Complete the page as follows.

Field	Description
Document Type	Enter the name of a document type, or click **Select Document Type** and select a type in the list.
Tool Name	Enter the name of a tool, or click **Select Tool** and select a tool in the list.

Field	Description
Rights	Select the rights to award user group members: • **Administrator**—Lets the user show, hide, move, edit, and delete tools that they create. (Tool creation is the same as placing an annotation on a MarkView document.) • **Optional**—Hides the tool by default. Lets the user show, but not move, edit, or delete the tool. • **Default**—Displays the tool by default. Lets the user hide, but not move, edit, or delete the tool. • **Mandatory**—Makes the tool always visible. The user cannot hide, move, edit, or delete it. • **None**—Hides the tool always. Does not let the user show, move, edit or delete the tool.

6. Click **Save**.

Chapter 6

MarkView Bar Code Generator

MarkView Bar Code Generator creates bar codes images that direct invoices being entered into the system to the correct workflow queue. For example, an employee can create a bar code cover sheet that identifies a document as a PO Invoice, scan the cover sheet with the invoice, and have MarkView send the scanned image to the PO Invoice Entry queue.

The employee can print the image. However, Bar Code Generator produces a GIF or PNG graphic that is viewable from any web browser.

You can reference MarkView Bar Code Generator from any web browser with a URL in the format:

http://host.domain:port/<base_path>/markview.barcode[?parameter=value]

You can insert a bar code image into any HTML document with the following syntax:

<IMG SRC="http://host.domain:port/<base_path>/markview.bar code[?parameter=value]>

Bar Code URL parameters

Configure generated bar codes by adding parameters to the URL that calls the bar code generator.

Parameter	Description
text	Specifies the value rendered as a bar code. • Default: NO DATA • Valid range: 1 – 100 characters, and limited to the bar code family of characters that you choose in the bar_format parameter. This is the only parameter required when you use the MarkView Bar Code Generator. Any passed characters are subject to the URL CGI parameter format. For example, to pass a plus sign, use the parameter %2B. The parameter **?text=1%2B3** displays **1+3**. The parameter, **?text=100%25+of+the+goal** returns **100% of the goal**. Invalid characters cause the following error message: "Illegal Character in Bar Code. Contact technical support." To use multiple bar codes on the same page, repeat the text parameter. For example, the following creates two bar codes on the page: ?text=hello&text=bar code2&text=bar code3
bar_width	Specifies the pixel width of the narrowest bars in the bar code. • Default: 2 for code 128, 2 for code 3 of 9 • Valid range: 1 – 5 pixels

Parameter	Description
bar_height	Specifies the pixel height of the bars in the bar code. • Default: 75 • Valid range: 1 – 900 pixels
img_width	Specifies the pixel width of the PNG or GIF containing the rendered bar code. • Default: the minimum width required to view the bar code • Valid range: 175 – 1000 pixels
img_height	Specifies the pixel height of the PNG or GIF containing the rendered bar code. • Default: the minimum width required to view the bar code • Valid range: 1 – 1000 pixels
x_offset	Specifies the horizontal origin of the bar code within the PNG or GIF in pixels. • Default: 20 • Valid range: 1 – 500 pixels
y_offset	Specifies the vertical origin of the bar code within the PNG or GIF in pixels. • Default: 20 • Valid range: 1 – 500 pixels
checksum	Specifies whether a checksum should be generated for the bar code. • Default: T • Valid values: [F][False][false][No][no][N][T][True][true][Yes][yes][Y] When using Kofax Capture, set checksum to F [False].
caption	Specifies whether or not a caption should be displayed below the bar code. • Default: T • Valid values: [F][False][false][No][no][N][T][True][true][Yes][yes][Y].

bar_format	Specifies the type of bar code to create.
	• Default: 128
	• Valid values:
	  • 39: Selects bar code type 3 of 9
	  • 128: Selects bar code type 128

spacing	Specifies the distance between the end of one bar code and the start of another. Use this parameter if you have multiple bar codes on the same page. • Default: 20 • Valid range: 0 – 1000
alignment	Specifies the alignment of the bar codes when you use multiple bar codes on the same page. This parameter determines the alignment of the bar codes relative to each other, not relative to the page. • Default: LEFT • Valid values: [LEFT], [CENTER], [RIGHT]

Parameter	Description
orientation	Specifies the orientation, or positioning, of the bar code. A value of VERTICAL rotates the bar code 90 degrees. The orientation occurs after the bar code is generated. A bar code with a height of 100 and a vertical orientation would result in a bar code with a width of 100. • Default: HORIZONTAL • Valid values: [HORIZONTAL], [VERTICAL]

Use Bar Code URL parameters

You only need to provide the text parameter for your bar code URL. The default values are sufficient for most implementations of MarkView Bar Code Generator. A typical bar code URL might look like this:

```
<IMG SRC="http://host.domain:port/mvasbcg/barcode[?parameter=value]">
```

To use multiple parameters in a bar code, use an ampersand (&) to join multiple parameters. For example:

```
<IMG SRC="http://170.test:777/mvasbcg/barcode?
text=Hello&bar_width=2&bar_height=20">
```

This example generates a bar code that says "Hello." The thin bars in the bar code example are 2-pixels wide and 20-pixels high.

Configuration parameters

Configure MarkView Bar Code Generator parameters by editing the orion-web.xml configuration file that is created when you first start the application server instance for the bar code generator.

Parameter	Description
BCG_DISPATCH_TIMEOUT	Specifies how long, in seconds, a temporary dispatcher remains unused before shutting down. If activity resumes, the time-out limit restarts. Default: 300 seconds
BCG_FILE_FORMAT	BCG_FILE_FORMAT Specifies the format of the picture file returned by the bar code generator. • Default: GIF (recommended value unless you require very small file sizes) • Valid values: PNG, GIF
BCG_INITIAL_DISPATCHERS	Specifies the minimum number of bar code generator instances running. Instances do not disconnect from the server or exit until the Application Server exits. Default: 3

Parameter	Description

BCG_LOG_LEVEL	Controls the logging output of the MarkView Bar Code Generator (set at the system level or user level).
	• Default: 1
	• Valid values:
	  • 1: Records only errors (sufficient unless you are debugging the MarkView Bar Code Generator installation)
	  • 10: Records extensive logging information, including debugging information
	The values 2 through 9 are not currently used.

BCG_MAX_DISPATCHERS	Specifies how many temporary dispatchers the bar code generator can create to meet increased demand. Temporary dispatchers pick up excess demand when the other dispatchers cannot handle the load. Default: 5

Specify the number of Bar Code packet retries

A bar code packet includes receipts with a bar code cover page. If MarkView receives a bar code packet that does not match an expense report work item, MarkView retries the matching process. MarkView preferences control the number and frequency of retries:

- MVEXP_UNMATCHED_PACKET_RETRY_COUNT: Specifies how many times MarkView tries to match a packet with an expense report before disposing of the packet.
- MVEXP_UNMATCHED_PACKET_RETRY_PERIOD: Specifies how often, in hours, MarkView retries the matching process.

Chapter 7

Export Server application administration

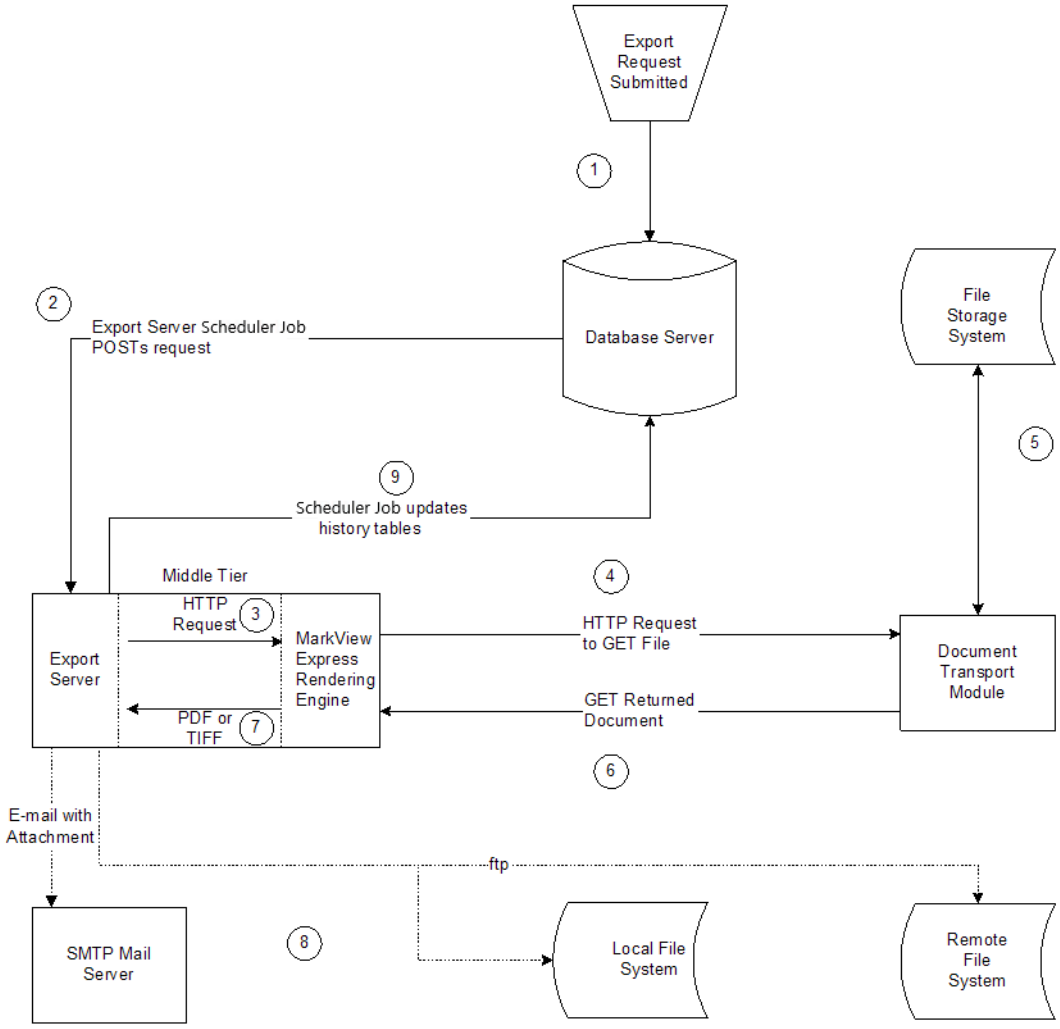
Use the information in this chapter to administer MarkView Application Server Export Server to ensure that print request process properly.

About the Document Export process

Export Server communicates with Mail Server using the SMTP protocol. If a firewall sits between Export Server and Mail Server, configure the firewall to allow SMTP traffic from Export Server. Although DTM and Export Server might use HTTPS, email encryption is incumbent on Mail Server. Mail Servers do not usually encrypt email.

The following numbers correspond to those in the Document Export Request Flow Diagram. The figure shows how Document Export completes an export request.

1. A request to email a document as an attachment arrives, or to write a document to the local file system or FTP it to a remote file system. The request can specify that the document be in PDF or TIFF format.
2. An Export Server Scheduler Job handles the request and posts an HTTP request to Export Server on the application server.
3. Export Server calls the viewer rendering engine, which is also running on the application server.
4. The rendering engine submits an HTTP request to DTM to GET the file.
5. DTM retrieves the document by direct file system access.
6. DTM returns the document to the rendering engine GET.
7. The rendering engine generates the PDF or TIFF and returns it to the Export Server GET.
8. Export Server fulfills the request by creating and sending an email with attached document, or by entering the document into a file system.
9. Export Server reports the request status to the database job, which updates the database history tables.



Document Export Request Flow Diagram

Start Document Export server tasks

The installation process creates the default Export Server Scheduler Job. If the job is stopped or you configure new queues, you must create start-stop jobs.

1. Verify that the Export Server Scheduler Job is available to poll for export requests.
2. Ensure that the application server instance for the Document Export J2EE application (viewer) is running.

If the Scheduler Job polls for requests, but the application server instance is not running, the requests are queued.

If the Scheduler Job is unavailable, incoming requests generate errors. Resubmit the requests after creating the Export Server job.

To create a Scheduler Job, see Create Export Server Scheduler Job on page 140.

Process print requests

To process print requests, start the Export Server J2EE application. The Export Server J2EE application, starts or stops when the application server starts or stops.

Monitor print requests

Track pending print requests from the Print Requests tab and the Export Requests tab in the MarkView Administration.

Export print requests that appear in the Print Requests list also appear in the Export Requests list. The export request type appears as a Print type to distinguish it from other export types, such as email and file system exports.

For upgrades, print requests can be served by the MarkView Print Server. The Request Type column in the Print Requests list view denotes whether a print request is queued for service by the Print Server or by the Export Server. Filter on a print queue or an export queue for pending requests.

1. Log in to MarkView and navigate to **Administration** > **MarkView Admin**.
2. Do one of the following:
   - To track Print requests, select the **Print Requests** tab.
   - To track Export requests, select the **Export Requests** tab.

Troubleshooting

Problems trying to email PDF documents are often related to permissions. For example, the configured servlet user name does not have the proper view and document type authorizations.

To test, ensure that the export user can open and view the document being exported. If this user cannot authenticate or does not have permission to view the document, the export operation fails. The export user must be able to log in to MarkView.

Chapter 8

MarkView Process utilities

This chapter describes the MarkView Process utilities that monitor and control your workflow. The utilities installed with your system depend on site configuration.

Administer MarkView Process Manager

The MarkView Process Manager is the workflow processing engine that pushes items through the MarkView suite workflow. The process manager is integrated into the database. Each instance of the manager runs independently, enabling you to stop and start one manager instance without affecting others.

Initialization parameters

The following database initialization parameters affect how database manager jobs run. Apply this information when you start the MarkView Process Manager.

Depending on the environment, a company might have more than one manager running, each with a different Manager ID.

The job_queue_processes parameter indicates the number of background processes that Oracle must start to handle jobs. (Required)

Set this parameter to a value greater than zero. Specify this parameter in the init.ora file of your DBMS.

Use the alter command from a PL/SQL prompt to set a database parameter.

For more information, see the Supplied PL/SQL Packages Reference for your version of the Oracle database.

MarkView Process Manager Job

The MarkView Process Manager runs as a job on the Oracle DBMS_SCHEDULER facility. Once started, the manager runs whenever the database starts. You can write a log file to the database server in a directory you create. To start a MarkView Process Manager job, invoke the SF_MGR.StartManager procedure that plans and runs a DBMS_SCHEDULER job. See procedure specification below.

```
procedure StartManager (
      JobName           OUT Varchar2,
      ManagerID         IN  Number,
      LogDirectory      IN  Varchar2,
```

```
      LogFilename       IN  Varchar2,
      PollingInterval   IN  Number Default 60,
      StartTime         IN  Date  Default sysdate,
      UseFileLogging    IN  Boolean Default True,
      CommitWork        IN  Boolean Default True,
      EnabledJob        IN  Boolean Default True
         );
```

The parameters are as follows:

Parameter	Description
JobName	The unique scheduler job name assigned to this job. It is generated as SF_RUN_MANAGER_<ManagerID> job. Use this name to track the job, stop the job, or delete the job.
ManagerID	The Manager ID assigned to this manager instance in MarkView Process Administration.
LogDirectory	The directory on the database server to which the log file should be written. Create this using the create directory command. Grant read and write privileges to this directory.
LogFilename	The name of the log file.
PollingInterval	The number of seconds MarkView Process Manager waits between attempts to look for events to process. Default value is 60.
StartTime	The exact time for this job to start. If you do not provide a value for this parameter, the job starts immediately.
UseFileLogging	Indicates whether to log messages to the file specified by LogDirectory and LogFilename.
CommitWork	Indicates whether this job is saved as soon as the procedure finishes. If this parameter is set to FALSE, the job is not saved until you explicitly perform a COMMIT.
EnabledJob	Indicates whether this job is enabled and will be run by default when created. If this parameter is set to FALSE, the job will not run until you explicitly enable it.

The default MarkView Process Manager is created by MarkView installer and named 'Main Manager'. Process Manager Job for the 'Main Manager' is started by default when MarkView is installed or updated. To administer the current Process Manager or to add a new one, go to Start MarkView Process Manager section.

Start MarkView Process Manager

Before running MarkView Process Manager, create a directory on the database server for log files.

1. Connect to sys as sysdba.
2. Create the directory, using an appropriate name, for example, PMLOG:

   ```
   create or replace directory PMLOG as '/usr/tmp';
   ```

   The system responds with a directory created message.
3. Grant read privileges on the directory:

   ```
   grant read on directory PMLOG to public;
   ```

   The system responds with a grant succeeded message.

4. Grant write privileges on the directory:

   ```
   grant write on directory PMLOG to public;
   ```

   The system responds with a grant succeeded message.
5. Run MarkView Process Manager. Log in to SQL*Plus using the MarkView schema user name and password and run the following from a PL/SQL prompt:

   ```
   set serveroutput on size 1000000
   declare
     job_name user_scheduler_jobs.job_name%type;
   begin
     SF_Mgr.StartManager(
       JobName => job_name,
       ManagerID => 1, -- change to an alternative Manager ID if needed
       LogDirectory => 'PMLOG',
       LogFilename => 'sfmgr.log',
       PollingInterval => 60,
       StartTime => sysdate,
       UseFileLogging => true,
       CommitWork => true);
     dbms_output.put_line('Job Name: ' || job_name);
   end;
   /
   ```

   The SF_Mgr.StartManager procedure creates a new Scheduler Job named 'SF_RUN_MANAGER_<Manager ID Number>' and returns the name of the started job into the `job_name` variable.

Test MarkView Process Manager

To view all currently active managers, go to the PL/SQL prompt and enter `select * from user_scheduler_jobs;`

The query returns all active Scheduler Jobs. Filter Process Manager Jobs by '%sf_mgr.RunManagerOnce%' job_action.

To test for failed jobs, enter `select job_name,job_action,enabled,state,failure_count from user_scheduler_jobs;`

The query returns job names that you can use to stop failed jobs.

Stop MarkView Process Manager

When you stop a manager, the manager completes the current job and shuts down.

To stop a manager, go to a PL/SQL prompt and enter the following:

```
begin
     dbms_scheduler.drop_job('SF_RUN_MANAGER_&managerid');
    end;
    /
   commit;
```

Where `managerid` identifies the manager ID to stop.

Stop an active MarkView Process Manager

If your manager enters into an infinite loop and cannot stop properly, identify and stop the manager directly.

1. Connect to sys as sysdba. From a PL/SQL prompt, run the following query to determine the SID of the manager: `select session_id from dba_scheduler_running_jobs where job_name = 'SF_RUN_MANAGER_&managerid';`
   Where `managerid` identifies the manager ID to stop (1 is the default value).
   The database returns the SESSION_ID (SID), for example:

   ```
   SESSION_ID
   ----------
          121
   1 row selected.
   ```

2. Using the SESSION_ID (SID) returned by the previous query, run the following query.
   `select serial# from v$session where sid = SIDnumber`
   The database returns the serial number of the manager, for example:

   ```
   SERIAL#
   ----------
            1
   1 row selected.
   ```

3. Using the SID and the serial number of the manager, run the following statement: `alter system kill session 'SID,Serial#';`
   For example: `alter system kill session '121,1';`
   The manager shuts down immediately and you receive a message that the system is altered.

Administer Process Event Generator

The MarkView Process Event Generator creates time-based events to use in a workflow. For example, to see how many document images load in an hour, use the hourtick event in a workflow.

The following event generators are available:

- SecondTick: Generates a secondtick event once per second
- MinuteTick: Generates a minutetick event once per minute
- HourTick: Generates an hourtick event once per hour
- DayTick: Generates a daytick event once per day

Start MarkView Process Event Generator

From a PL/SQL prompt, enter the following statement:

```
set serveroutput on size 1000000
declare
  job_name user_scheduler_jobs.job_name%type;
begin
  sf_mgr.Generate<IntervalTick>Events(
    JobName=>job_name,
    UserID=>'ADMIN',
```

```
    CommitWork=>True);
  dbms_output.put_line('Job Name: ' || job_name);
end;
/
```

Replace <IntervalTick> with the event interval to measure. For example, the following generates an event once per hour:

```
set serveroutput on size 1000000
declare
  job_name user_scheduler_jobs.job_name%type;
begin
  sf_mgr.GenerateHourTickEvents(
    JobName=>job_name,
    UserID=>'ADMIN',
    CommitWork=>True);
  dbms_output.put_line('Job Name: ' || job_name);
end;
/
```

Test MarkView Process Event Generator

To see all the currently active event generators, run a query against the user_scheduler_jobs view.

From a PL/SQL prompt, enter `select * from user_scheduler_jobs;`

The query returns all active jobs. In this list, you can find all active event generators filtering jobs by '%sf_client.AlertEvent%' job_action.

Stop MarkView Process Event Generator

From a PL/SQL prompt, enter the following:

```
begin
  dbms_scheduler.drop_job('&jobname');
 end;
 /
commit;
```

Where `jobname` identifies the Process Event Generator job to stop.

Administer Connector Workflow Manager

MarkView Installer plans and runs scheduler job for the default MarkView Connector Workflow Manager 'Main Manager'.

To administer the current Connector Workflow Manager or to add a new one, go to Start Connector Workflow Manager Job section.

Start Connector Workflow Manager Job

To create a database job for Connector Workflow Manager in the MarkView schema, execute the following PL/SQL script:

```
set serveroutput on size 1000000
```

```
declare
  job_name user_scheduler_jobs.job_name%type;
begin
  MVCN_Workflow.StartManagerJob(
    JobName => job_name,
    ManagerID=> 1, -- change to an alternative Manager ID if needed
    LogDirectory=> '<logging_directory>',
    LogFilename => 'mvcn_workflow.log',
    UseFileLogging => TRUE,
    CommitWork=>TRUE);
    dbms_output.put_line('Job Name: ' || job_name);
end;
/
```

where `logging_directory` is a directory that already exists on your database server.

The database Administrator must configure the database with job-write access to this directory. See your Kofax MarkView Installation Worksheet.

The unique scheduler job name assigned to this job is generated as 'MVCN_WF_RUN_MANAGER_<ManagerID>'. Use this name to track the job, stop the job, or delete the job.

Test MarkView Connector Workflow Manager Job

To view all currently active connector workflow managers, go to the PL/SQL prompt and enter `select * from user_scheduler_jobs;`

The query returns all active Scheduler Jobs. Filter Connector Workflow Manager Jobs by '%mvcn_workflow.StartManager%' job_action.

To test for failed jobs, enter `select job_name, job_action, enabled, state, failure_count from user_scheduler_jobs;`

The query returns job names that you can use to stop failed jobs.

Stop MarkView Connector Workflow Manager Job

When you stop a manager, the manager completes the current job and shuts down.

To stop a manager, go to a PL/SQL prompt and enter the following:

```
begin
dbms_alert.signal('170$MANAGER_ALERT','Stop Manager:');
dbms_scheduler.drop_job('MVCN_WF_RUN_MANAGER_&managerid');
end;
/
commit;
```

Where `managerid` identifies the manager ID to stop.

Stop an active MarkView Connector Workflow Manager Job

If your manager enters into an infinite loop and cannot stop properly, identify and stop the manager directly.

1. From a PL/SQL prompt, run the following query to determine the SID of the manager job: `SELECT session_id FROM dba_scheduler_running_jobs where job_name = 'MVCN_WF_RUN_MANAGER_&managerid';`

   Where `managerid` identifies the manager ID to stop (1 is the default value).

   The database returns the SESSION_ID (SID) , for example:

   ```
   SESSION_ID
   ----------
          121
   1 row selected.
   ```

2. Using the SESSION_ID returned by the previous query, run the following query.

   `select serial# from v$session where sid = SIDnumber`

   The database returns the serial number of the manager, for example:

   ```
   SERIAL#
   ----------
            1
   1 row selected.
   ```

3. Using the SID and the serial number of the manager, run the following statement: `alter system kill session 'SID,Serial#';`

   For example: `alter system kill session '121,1';`

   The manager shuts down immediately and you receive a message that the system is altered.

Chapter 9

Maintain DTM

MarkView Document Transport Module (DTM) provides a secure means to transport image information from your database to the document server.

Use the information in this chapter to modify the authorization method that the system uses. For basic DTM setup, see *MarkView Administrator's Guide, Volume 1.*

DTM daily maintenance

MarkView DTM requires minimal daily maintenance. If you experience performance issues or update the software, you can test and reset DTM. Otherwise, you do not need to perform any administrative tasks.

DTM has an internal status check, which reads preferences, connects to the database, and describes the procedure in the database. The status check reports the status to the user and requires no authentication.

Test DTM

1. In the browser window, enter the URL `http://server:port/mvasdtm/markview?action=test`.

   This URL tells DTM to perform the status check. If DTM encounters errors, a message similar to the following appears:

   ```
   Internal error An internal error occurred during execution.
   "Initialization Failed."
   Please contact the administrator.
   ```

2. In MarkView Administration, set the MVAS_DTM_LOGGING_LEVEL preference to DEBUG and retry the test.

   The application server output log provides information about the specific error. If the test finds no errors, information similar to the following appears:

   ```
   MarkView Document Transport Module <version>

   Copyright <year> Kofax. All rights reserved.

   Uptime
   Running since: <DateTime>
   Total service requests: 5

   Current Load
   Current service requests: 1
   Open database connections: 0
   ```

```
JVM Memory
Type  Kilobytes (KB)
Free  449003
Total  518592
Maximum  1036928

Testing database connection...success. Connected to MARKVIEW@vis01

Diagnostic Information

Request Information
getAuthType(): null
getContextPath(): /mvasdtm
getPathInfo(): null
getPathTranslated(): null
getQueryString(): action=test
getRequestURI(): /mvasdtm/markview
getRequestURL(): http://r4ebsr12.kofax.com:7777/mvasdtm/markview
getServletPath(): /markview

HTTP Headers
ACCEPT=text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
ACCEPT-CHARSET=ISO-8859-1,utf-8;q=0.7,*;q=0.7
ACCEPT-ENCODING=gzip,deflate
ACCEPT-LANGUAGE=en-us,en;q=0.5
CONNECTION=keep-alive
COOKIE=__utma=164137727.2574543596640978000.1247693675.1247693675.1247693675.1;
       __utmz=164137727.1247693675.1.1.utmcsr=(direct)|utmccn=(direct)|
utmcmd=(none);
       oracle.uix=0^^GMT-4:00^p
HOST=r4ebsr12.kofax.com:7777
KEEP-ALIVE=300
USER-AGENT=Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.14)
 Gecko/2009082707 Firefox/3.0.14 (.NET CLR 3.5.30729)
ORACLE-ECID=1255990025:10.60.1.199:9435:0:156,0

Request Parameters
action=test
```

DTM security

MarkView DTM provides layers of security for document access.

The first layer is user name authentication. Based on the user name and password supplied by the user/browser, authorization is granted or denied. The simplest form of user name authorization is realm-based authentication. Realms, however, have a limited scope and are browser-specific. DTM provides cookie authentication as an alternative. Once the user name and password are authorized, the user must also have document-level authorization.

MarkView privileges affecting security

In MarkView, the privileges assigned to a MarkView user group provide a fine level of control over which document types a user group can access. From MarkView Administration, you can set the privilege through document type or user group maintenance.

For example, if the Invoice Entry user group has the Access Documents privilege set to Yes for the document type “US Invoice,” group members can view US Invoice documents in MarkView Viewer.

If the privilege is set to No, users in this user group receive an error if they attempt to open a US Invoice document in the viewer.

The following rules apply:

- The system checks the privilege only if the MarkView suite preference SECURITY_ENFORCE_ACCESS_CONTROL is set to TRUE.

  If FALSE, all user groups can access all document types regardless of the privileges set. If the preference is missing or set to an invalid value, MarkView assumes a setting of TRUE.
- If the user group has no privilege set for a particular document type, MarkView assumes a value of NO.
- If a user belongs to many user groups and one user groups has authorization to access a particular document type, the user can access the document type.
- Setting a privilege for All Document Types applies the setting to all document types.
- Setting a privilege for the All Users user group, applies to all members of the All Users group. However, not all users are automatically members of the All Users group.
- When checking a privilege, MarkView also checks that the document type is enabled as of the current date/time. If the document type is disabled, no users have access to the document type.
- The DTM checks that the current user has the required privilege for the requested operation (Upload, Download, Delete). The relevant MarkView privileges include:
  - MarkView Web Folder Upload
  - MarkView Web Folder Download
  - MarkView Web Folder Delete

About realms

When a user connects to a secure application server, the system prompts for a user name and password. The window displays a message that says “Realm xxxx” or “Enter username for xxxx at”, depending on the browser. The xxxx represents the realm, which is the web space against which the user is being authenticated. If the user is on the same server, but authenticating against a different web space, the browser reprompts for a user name and password.

The browser stores the user name and password combinations, which the web server cannot access. When the browser requests a URL, it looks to see if it has a corresponding user name and password for the same web server and the same web space. If it does, the browser sends the user name/password combination with the request. If the browser does not, it prompts for the information.

To configure your realm, set the application server preference DTM_SECURITY_REALM in the web.xml file to an appropriate value.

The application returns the exact string, which eliminates reprompting for user name and password.

When reprompting for a user name and password is unacceptable, consider installing a cookie in the user's browser to be read for security instead.

The system sends the user name, password, and realm to the Authorized function for validation.

Validate users through cookies

Cookies provide the most flexibility for validating a user name and password. The application controls the cookies and the browser stores them. This makes validation easier when transferring from one realm to another or when multiple applications with different validation methods are involved.

DTM takes two approaches to validating users.

- DTM first tries the basic method of validation, using standard user name, password and realms.
- If the first approach fails, DTM tries to read a cookie by looking for the parameter UserCookieName. If the parameter exists, DTM retrieves the value, uses the value to look up the corresponding cookie, and gets the cookie value.

For example, if the common gateway interface (CGI) passes

?UserCookieName=myCookieName

1. DTM looks at the CGI value, which is myCookieName.
2. DTM looks at the value of the cookie myCookieName.
   If the cookie value is set to JSMITH, DTM uses JSMITH as the name to authenticate.
3. DTM sends the name to the Authorized function to validate the user name.

You can use the following parameters:

- userCookieName: The user name passed into the Authorized function
- passwordCookieName: The password passed into the Authorized function
- realmCookieName: The realm passed into the Authorized function

Appendix A

Troubleshooting

This chapter describes basic troubleshooting and answers the following questions:

- How do I verify that MarkView and the viewer are running?
- How do I find an invoice that seems to be lost in MarkView?
- Why can't a MarkView user send an invoice the another MarkView user for comment?
- Why doesn't a MarkView user have access to a particular queue?
- Why does a MarkView user receive a "no items pending" message when MarkView has items pending?
- How do I recognize and handle batch class issues when using Kofax Capture?
- How do I address print or email issues?
- What are the support tools and what do they do?
- Why are invoices routed to workflow admin?

If you encounter a problem with a MarkView component, use the information in this chapter before contacting Kofax Technical Support. For information about accessing the latest support tools, see Getting help with Kofax products on page 9.

Solve issues with user functions

If MarkView User Groups and Roles are set up incorrectly, your users might encounter problems accessing or viewing MarkView documents and interface elements. For example:

- A menu item is not available in MarkView home.

  In Module Admin, check the Menu Item Group Auth table and confirm the user group assigned to the menu. In Markview Admin, verify that the user is a member of that group.
- An AP processor receives a "no items pending" message even when MarkView has items pending.

  In Process Administration, verify that the user's role has the retrieval privilege for the queue and that work items are pending for the user.
- A Markup is not available in the viewer.

  In MarkView Administration, check the Tool Privilege Auths for the missing tool. Ensure that the user is assigned to a group that has access to the tool.
- A user cannot see work items in the Process Monitor.

  Use Process Administration to ensure that the user has roles enabled that have the examine privilege authorized for the Process Monitor.

Identify common AP Entry mistakes

- Clicking Get Next too frequently

  Causes unattached documents to end up in the AP Working Folder or Console
- Placing transitional actions too early

  Causes an error message because the work item is not attached to a header; for preapproved, the item is not fully distributed or matched to a PO.
- If no transitional action is taken

  Causes the invoice to remain in the Working Folder or Console.

Resolve email and print issues

Email

Problems trying to email PDF documents can be related to permissions. For example, the configured servlet user name does not have the proper authorizations.

To test, ensure that the export user can open and view the document being exported. If this user cannot authenticate or does not have permission to view the document, the export operation fails. The export user must be able to log in to MarkView.

Print (for WebLogic 12.1.3 only)

With MarkView Viewer on the WebLogic 12.1.3 application server, the following error appears in the log file when you try to generate a print request to the server:

```
java.awt.print.PrinterException: No print services available
```

To correctly configure the printer feature, do the following:

1. Log in to the WebLogic Server Administration Console.
2. In Domain Structure, expand **Environment** and select **Servers**.
3. In the Configuration tab, select **markview_server**.
4. On the Settings for markview_server page, select the **Server Start** tab.
5. In the Arguments field, add the following to the end of the line:

   ```
   -DUseSunHttpHandler=true
   ```
6. Save changes and restart the server.

Investigate workflow issues

- If you are unsure as to why invoices are routed to a particular queue, for example, workflow admin, check the preference settings and selections made during Organization Setup. To

determine what caused a work item to be routed to the Workflow Admin queue, use the Process Monitor to view the Work Item History. The last transition includes the error that caused the work item to be routed to the Workflow Admin queue.

- A user name changed in an Oracle HR record
  - Work items assigned under the original user name cannot be accessed.

    To solve this issue, assign the new user name in MarkView as the alternate to the original user name. (MarkView retains both user names.)
  - Work items that are in-process and need approval at the next level in the hierarchy are routed to the Workflow Administration queue. This occurs because the system cannot determine the next Approver in the hierarchy.

    To solve this issue, for the new MarkView user name, change the MVERP_USE_ALTERNATE_AUTHORITY preference setting at the user level to **ALL** and route the affected work items back to the previous queue. Once all work items that were put into process under the original user name finish processing, you can change the preference setting.

Address performance issues

- If your site experiences performance slow-downs and sporadic Java Virtual Machine (JVM) crashes preceded by a slow-down, try restarting the MarkView Application Server on a weekly basis. Doing so refreshes the runtime environment.
- If your site processes large batches and experiences excessive memory usage that result in performance slow-downs, verify that the MarkView preference SPLIT_DOCUMENTS is set to TRUE (default).

  When set to FALSE, this preference instructs the connector to combine all pages from all images into a single multi-page image, which leads to excessive memory use.

Verify the state of MarkView and the viewer

1. Log in to MarkView and navigate to **Administration** > **Verify MarkView**.
2. Click one of the show-document icons.

   MarkView Viewer opens and displays the MarkView icon or a blank document. Both MarkView and the viewer are running if the document image opens in MarkView Viewer.

MarkView Viewer displays an error about the number of pages

If the viewer displays an error that the number of pages should never be less than 1, check the viewer workstation settings as follows.

1. Log in to MarkView and navigate to **Administration > MarkView Admin**.
2. Select the **Workstations** tab.
3. Locate **MarkView Express** and click **Details**.
4. In the Details window, verify that the Serial Number is **EXPRESS**. Any other entry in this field prevents the viewer from functioning.
5. Change the field entry to **EXPRESS** and click **Save**.

Find a lost invoice

Use the Process Monitor to locate lost invoices. For information about using the Process Monitor, see the *Kofax MarkView Administrator's Guide, Volume 1*.

Server connections

Without a connection to the database and servers, MarkView Viewer cannot function properly. Test your server connections to make sure that all components are communicating properly.

Connection between user machines and servers

Test the connection between the machines running the viewer and the following servers:

- MarkView Application Server
- MarkView Document Server

Use the ping command to test the connection. If a ping fails or takes unusually long, there is an issue on your network. Run ping from a user machine: **ping computername**

For example, to test the connection from a user machine to the server machine test.kofax.com, issue the command **ping test.kofax.com**.

If your user machines are on multiple subnets, run ping from machines on each of the different subnets.

The MarkView suite transmits TIFF files that have an average size of approximately 50 kilobytes. For a truer test of the document server connection, run ping with a packet size of 50 kilobytes. For example, from a command prompt, you might issue the command **ping -l 50000 test.kofax.com**.

A packet size of 50 kilobytes might cause ping to time out, which gives a false impression of a lost packet. Raise the time-out level to compensate when testing a large packet size.

Windows performs four pings and stops. UNIX continues to ping the target computer until you press Ctrl-c.

Evaluate the results in terms of the following:

- How long did the ping take? For computers on the same network, ping times of 30 milliseconds or less are normal. For computers running on separate networks or over VPN connections, ping times may be significantly longer. A ping time of over 500 milliseconds may indicate that a slow network connection is responsible for the poor performance of MarkView Viewer. A short ping time, although not definitive, is a good indicator of overall network health.
- Were any packets lost? On a healthy network, this should rarely happen. Occasionally, a packet may appear to be lost, when in fact the ping timed out. Raise the timeout period and rerun the test.
- Were the results from one subnet significantly different from other subnets? If so, your network administrator might need to take a detailed look at the LAN topology or at the routers.

A slow response or lost packet could indicate any of the following problems:

- The network connection between a user machine and the specified server is down.
- The network connection is slow, which may be related to the network architecture. See the *MarkView Planning Guide* for details about the recommended MarkView suite architecture.
- The network connection is reasonably healthy but you are testing during peak time when the network is very busy (actually, a good time to test). Rerun the test off-peak to determine the difference between peak and off-peak performance.

Time how long the viewer takes to display an image

Time how long it takes for the viewer to open after you request a document. Time how long it takes for the image to appear. Displaying the image typically involves Document Transport Module (DTM). If performance is sluggish, perform a database view on the page and document ID requested, as follows:

```
select platform_name, complete_filename
from mv_page_image_paths
where document_id = document-ID
and page_number = 1;
```

This returns the complete string the viewer uses to retrieve the TIFFs for the document. The default platform is HTTP_TO_DTM. Navigate to the returned URL to test DTM connectivity and speed.

Also do the following:

- Ping the document server.
- Time an FTP event to the document server.
- Set the log level to debug (level 10) and examine the DTM log. Pay particular attention to significant gaps in time-stamped events.

Improve performance

To further improve performance, follow the guidelines for clustering and load-balancing provided by your application server supplier.

View type authorizations

Do not authorize any user group to modify Queue Status Markups views. Doing so can lead to inadvertent loss of data.

The underlying code in MarkView Toolkit database objects (MVT_Queue_Status) deletes markups placed on a Queue Status Markup view each time someone views the document. This setup allows the Queue Status Markups for each document view to show the status of the current queue.

If a group can modify the Queue Status Markup view, someone could inadvertently put it in the current view, proceed with markup placement, and assume that the updated document is preserved. However, the next time someone views the document, the markups are deleted as part of normal processing.

View Expense Reports after an AP import

If you import an AP Invoice and then purge Expense Reports, the Expense Report summary information still appears in Expense Home, with a link to View Receipts. When you click the link, however, you not see the receipt document because the Expense Report was purged from Oracle.

MarkView DTM issues

Upload large files using MarkView DTM issue

The MarkView DTM error logging file ($ORACLE_HOME/j2ee/<MV_INSTANCE>/log/mvdtm.log) tracks document upload failures. A typical message appears as follows:

```
Message: The uploaded file is too large. Please contact your
    administrator.
Log Message: Posted content length of 13196578 exceeds limit
    of 5120000
```

Images appear distorted running in a Terminal Server environment

In a Terminal Server environment, if images appear fuzzy or distorted, setting the monitor display to 16-bit (or higher) color should correct the problem. This also requires that you set Citrix to transmit 16-bit (or higher) color, as Citrix is frequently configured to transmit 8-bit color to preserve bandwidth.

Diagnose DTM issues

Before calling Technical Support, complete the following procedures to verify the system environmental and gather information to help diagnose the issue.

Verify the application server requirements

Verify that your application server meets the criteria specified in the *Kofax MarkView Planning Guide for Oracle E-Business Suite* and the *Kofax MarkView Technical Specifications* document on the Kofax MarkView Product Documentation site.

1. Verify the Java version:
   a. Log in to a shell as the user who installed the application server.
   b. Execute the following command:
      `java -version`
2. Verify that the system includes both a JDK and a JRE.
   a. Log in to a shell as the user who installed the application server.
   b. Execute the following command:
      `which javac`

3. Ensure that the JDK/bin directory appears in the PATH environment variable.
4. Confirm that all operating system patches needed to run the JDK were applied.
   Go to Java.sun.com to find patch requirements.
5. Record the environment variable settings for your platform by executing the appropriate shell command:
   `env > myenv.txt` (UNIX)
   `set > myenv.txt` (Windows)

Check the internal status

Navigate to **http://server:port/mvasdtm/markview?action=test** to perform an internal status check on the MarkView DTM.

Verify the database access

If the MarkView DTM log ($ORACLE_HOME/<application server_instance>/logs/mvdtm.log) indicates a database access problem, test the connection information:

Log in using the user name, password, and host connection string found in $ORACLE_HOME/<application server_instance>/config/data-sources.xml.

Verify pathnames

MarkView documents are stored as TIFF files in the Document Server file system. MarkView stores the pathnames of the TIFF files in the MarkView database in a view named MV_PAGE_IMAGE_PATHS. Check the work item detail window to see the path of a particular image.

1. Log in to MarkView Administration as the owner of the MarkView schema and retrieve the value of the MVAS_DTM_WORKSTATION_SERIAL_NO preference (typically, MVAS_DTM_1).
2. Log in to SQL*Plus as the MarkView schema owner and issue a command in the following form, using the value you retrieved in step 1.
   ```
   select platform_id from mv_workstation
   where serial_number='MVAS_DTM_1';
   ```
   The database returns the platform ID of the specified workstation. For example:
   ```
   PLATFORM_ID
   -----------
   5
   ```
3. In SQL*Plus, find a document ID that uses HTTP_TO_DTM or HTTPS_TO_DTM as a platform name by issuing the following command:
   ```
   select document_id from mv_page_image_paths
   where platform_name='HTTP_TO_DTM';
   ```
   The database returns the document ID that uses the specified platform. For example:
   ```
   DOCUMENT_ID
   ------------
   51
   52
   ```
4. In SQL*Plus, find the pathnames that the MarkView DTM accesses by issuing a command in the following format, using values that you retrieved in steps 2 (platform ID) and 3 (document ID):

```
select complete_filename from mv_page_image_paths
where document_id=51
AND platform_id=5;
```

The database returns the pathname of the specified document and platform. For example:

```
COMPLETE_FILENAME
-----------
/appl01/homes/qakk01/working_dir/middletier/docserver/0/0/5/4
 /00000054.tif
```

5. Log in to the MarkView DTM host as the application server user and copy the pathname returned in step 4 to a temporary directory. A successful copy indicates that the pathname is valid.

   If everything looks all right, set the MarkView DTM logging level to debug (set DTM_LOG_LEVEL to ALL).

   Have the additional information outlined in the following table available when you contact Technical Support.

Information Needed	Location
Log files • server.log • global-application.log • default-web-access.log	Logging Directory (from the *Kofax MarkView Installation Worksheet)*
application.log	<home_directory>/application-deployments/mvasdtm directory
Database preference values	All preferences matching the filter MVAS_DTM% in MarkView Administration
Application server preference values	The orion-web.xml file in the <home_directory>/application-deployments/mvasdtm/mvasdtm directory
Volume paths for OPEN volumes	In MarkView Administration: • Display the list of volumes (Volumes tab). • Click **Details** for an OPEN volume. • Select the **Volume Paths** subtab.

Document Export

If the Export Server Scheduler Job polls for requests, but the viewer application server instance is not running on the application server, the requests are queued. If the Export Server Scheduler Job is not available, incoming requests generate errors. You must resubmit the requests after creating the Export Server job.

A default job named MV_EXPORT_POLL_PROC_ALL_JOB for DEFAULT_EXPORT_SERVER runs automatically after you install MarkView. However, you may need to recreate this job, or create an additional Scheduler Job. Multiple jobs could be required if you use Print Server or additional

export queues. See Export Server application administration on page 118 for information about Print Server administration.

Create Export Server Scheduler Job

To create the Export Server Scheduler Job, call mv_export_admin.CreateJob, use code similar to or identical to the following:

```
declare
begin
-- Start up the Export Server.
mv_export_admin.CreateJob(
          JobName => 'MY_EXPORT_POLL_PROC_JOB',
          ServerName => 'MY_EXPORT_SERVER',
          StartTime => sysdate,
          PollingInterval => 5,
          CommitWork => true,
          UseFileLogging => true,
          LogNormalMessages => true,
          LogDirectory => '/tmp/log',
          LogFilename => 'MyExportServer.log');
end;
/
```

Where:

- MY_EXPORT_POLL_PROC_JOB is the name of the new Export Server Job created. Replace this value by your own job name.
- MY_EXPORT_SERVER is the exact name of your Export Server as defined in MarkView Administration. The default name is DEFAULT_EXPORT_SERVER
- /tmp/log is the log directory.

> Before you run this job ensure that the LogDirectory exists and that the database can write to this directory.

Because CreateJob is a wrapper around DBMS_SCHEDULER, the Export Server job restarts when the database restarts.

Confirm that Export Server Scheduler Job is running

Since CreateJob uses the DBMS_SCHEDULER infrastructure, run this SQL command:

```
select job_name, job_action, enabled, state, failure_count from
user_scheduler_jobs where job_action like '%mv_export_svr%';
```

Examine the output to determine if the Export Server job is still running. The output is similar to:

```
JOB_NAME
----------
JOB_ACTION
-----------------------------------
MY_EXPORT_POLL_PROC_JOB

begin
   mv_export_svr.PollAndProcessAll('MY_SERVER',
```

```
                                    UseFileLogging    => FALSE,
                                    LogNormalMessages => FALSE,
                                    LogDirectory      => '',
                                    LogFilename       => '');
end;
MV_EXPORT_POLL_PROC_ALL_JOB
begin
   mv_export_svr.PollAndProcessAll('DEFAULT_EXPORT_SERVER',
                          UseFileLogging => TRUE,
                          UseNormalMessages => TRUE,
                          LogDirectory   => 'TEMP_LOGS',
                          LogFilename    => 'mv_export_vis01.log');
end;
/
```

This example shows that the current user started two jobs. The relevant job calls the routine mv_export_svr.PollAndProcessAll and specifies the relevant servers (for example, MY_SERVER and DEFAULT_EXPORT_SERVER). In this case, the relevant job names are default MV_EXPORT_POLL_PROC_ALL_JOB and custom MY_EXPORT_POLL_PROC_JOB.

Terminate Export Server Scheduler Job

To terminate the Export Server Scheduler Job, call the mv_export_admin.RemoveJob procedure. This procedure takes a single argument, which is the job name. For example, to terminate the server job name MY_EXPORT_POLL_PROC_JOB, issue the following command:

```
begin
mv_export_admin.RemoveJob('MY_EXPORT_POLL_PROC_JOB');
end;
/
```

Capture and Output components

There Are No Messages in the MarkView Event Log

This applies only to MarkView Bar Code Server that runs as a Windows service.

If you encounter this condition:

- Reboot the system after creating a new service instance.
- The log may have become full. For the best results, set the MarkView Event Viewer property, when maximum log size is reached, to Overwrite events as needed.

Logging resources

This section provides pointers to logging resources available across MarkView.

Capture and Output components

As a Windows service, MarkView Bar Code Server uses the Event Viewer application for logging.

The log files for Import Server, Import Server API, Data Export Server, and Mail Gateway are located in the `<path>/logs` folder, where `<path>` is the location where you installed these components.

The debug logging level is enabled by default.

Applications log

The application server log files are located in the `<path>/log` directory, where `<path>` is the location where you installed MarkView.

For information about setting up the log files location, see the *Kofax MarkView Installation Guide*.

MarkView Self-Service Invoice (SSI), another Application Server application, is a separate product. The Self-Service Invoice log files are located at `<path>/<ssi_instance_name>/log`.

MarkView installation

The MarkView installer records up to four log files during installation. For information about these log files, see the *Kofax MarkView Installation Guide*.

Clear the log files

Log files can grow exponentially, which can degrade performance, and can be difficult to analyze when investigating issues. For the best results, perform regular maintenance on the log files.

Archived log files provide an historical record that allows you to establish a baseline, detect patterns, and so forth, but eventually they outlive their usefulness. Determine a suitable retention period after which you purge archived files.

1. Establish a volume threshold at which to roll over the log file (20 MB, for example).
2. When the log file reaches the threshold, stop the process or application that is generating the log.
3. Move the log file to an archive directory. Use a timestamp naming convention to identify the period of logging activity covered.
4. Later, go back and condense archived files (zip or tar) for more efficient storage and portability.

Oracle RDBMS

This section gives you information about the following:

- init.ora File
- Oracle DBMS Error Messages

Tune the init.ora file

The database initialization file, init.ora, contains parameters that can impact MarkView operations. The DBA should monitor performance for disruptions in service that may warrant adjustments to parameter values, such as the following:

- JOB_QUEUE_PROCESSES determines the number of background processes to start to handle database jobs (DBMS_JOB jobs and Oracle Scheduler jobs). Each job requires a process. Jobs for which no processes exist do not run.
- SESSIONS (and potentially SHARED_SERVER_SESSIONS) determines how many concurrent sessions can run on the database, and how many connections to support. Users can be denied a connection, for example when trying to access MarkView home, if no sessions are available. As each installation has different demands, there is no optimal or prescribed number. Usage statistics over time should provide some guidelines for setting this parameter. As a rule of thumb, each application server instance pools 10 sessions.

Oracle DBMS error messages

This section lists some common Oracle DBMS error messages. For a listing of all Oracle error messages, refer to Oracle MetaLink.

ORA-00001: unique constraint (string.string) violated

This error appears when users attempt to create a duplicate of a record or key that must remain unique. For example, you might want all of your users to have unique user IDs. If you attempt to create more than one "jsmith" user, you trigger this error.

ORA-01000: maximum open cursors exceeded

This error usually indicates that your Maximum Open Cursors parameter is set too low.

ORA-01401: inserted value too large for column

This error appears when you try to save an object name or description that is longer than the database allows. Create a shorter name or description and try again.

You can use the SQL describe command to determine the maximum allowable length for object names and descriptions. Describe returns descriptive information about the table and table columns.

A describe statement takes the form

```
describe table_name
```

For example, describe MV_USER_PROFILE

ORA-01403: no data found

Although a variety of conditions can trigger this error, two are the most common causes:

- If you recently ran a seed data script, and the script failed, you do not have a complete set of data and database objects. Check your log files to see if the seed data scripts ran successfully.
- If you recently tried to save data, and the save failed, you may be missing some data. Check your log files to see if all of your recent saves were successful.

ORA-01422: exact fetch returns more than requested number of rows

This error appears if you run the same seed script repeatedly. Performing multiple successful runs of the same seed script can create duplicates of the object. For example, you can create multiple

instances of the same tool objects. When the database then tries to access the object, it finds duplicate objects and generates an error. If you encounter this error, query the database to verify that you do not have duplicate objects. If you do have duplicate objects, delete the duplicates and verify that each object is unique.

ORA-01722: invalid data while updating table

This error occurs when commas appear in numeric data. Make sure that you filter out commas before entering form data into the database.

ORA-02292: integrity constraint (string.string) violated - child record found

This error appears if you try to delete an object that has child objects attached to it. Delete all of the child objects and try to delete the parent object again.

ORA-03113: end-of-file on communication channel

This error indicates that the Oracle client (user application) lost the connection to the database. The cause of this issue can be difficult to diagnose. Check the log files and test the connections of the various components of the MarkView suite.

ORA-06502 PL/SQL: numeric or value error: character string buffer too small

This error indicates that the contents of a text field in a form (or MarkView Accounting Details) is longer than the database can accept. Work around this problem by reducing the length of text entered into an entry field or form. The maximum number of characters allowed in a text field should be fewer than the character size of the database column receiving the input.

Use the SQL describe command to determine the maximum length allowed for object names and descriptions. The command returns a description of the table and table columns.

The describe statement uses the following form:

describe table_name

For example, describe AP_Users.

ORA-06512: at 'MARKVIEW.MVCN_VALIDATION', line xxx

Ensure that every MarkView user has a corresponding Oracle HR record with a valid email address. Users without email addresses cannot participate in MarkView workflows.

ORA-06550: line string, column string: string

This error usually appears as the result of a SQL syntax error. Make sure that your SQL scripts and packages are properly formed and are compiling properly. Once you check your scripts, repeat the action that caused the error.

ORA-20001: The user ADMIN has not been setup properly in Oracle Apps.
ORA-20999: Operation: Set Created By raised exception INSIDE method

Ensure that every MarkView user has a corresponding Oracle HR record with a valid email address. Users without email addresses cannot participate in MarkView workflows.

ORA-24247: network access denied by access control list (ACL)
ORA-29273: HTTP request failed

These errors appear when you did not add your host to the Access Control List. To solve this issue run the following statement as SYS:

```
BEGIN
DBMS_NETWORK_ACL_ADMIN.CREATE_ACL (
  acl => 'markview.xml',
  description => 'Permissions to access MarkView Middle Tier',
  principal => 'MARKVIEW', -->> add your markview schema name
```

```
  is_grant => TRUE,
  privilege => 'connect');
DBMS_NETWORK_ACL_ADMIN.ASSIGN_ACL (
  acl => 'markview.xml',
  host => '*.mycompany.com'); -->> add your company name
END;
```

Oracle Forms runtime errors

FRM-40654: Record Has Been Updated By Another User. Re-Query To See Change

You may see this error for the Oracle Quick Invoices form in Oracle EBS 12.x if you applied certain Oracle patches. For more information, see the *Kofax MarkView Installation Guide* description of enabling Kofax MarkView Quick Invoices. Follow the instructions in Step 11: Editing PRE_UPDATE, LOCK_ROW, and UPDATE_ROW Procedures.

FRM-40735: PRE-RECORD trigger raised unhandled exception ORA-06502

```
FRM-40735: PRE-RECORD trigger raised unhandled exception ORA-06502.
---------------------------------------------------------
ORA-01403: no data found
```

This is the result of a form compilation problem.

In MarkView for Accounts Payable you might see this error from **Invoices** > **Entry** > **Invoices** > **Actions** if you have an invoice listed.

In MarkView for Expense Management, you might see the form compilation error after opening the Working Folder and clicking the Open button, or from an Expense form, after clicking GetNext.

FRM-40819: System variable is not modifiable

You may see this error from Accounts Payable if you try to search by invoice number. This error may result from having compiled the modified Oracle form in Form Builder rather than the compiling the modified Oracle form from the command line as recommended. To fix the problem, recompile the modified form. For more information, see the *Kofax MarkView Installation Guide* description of enabling the Invoice Workbench form and the Vendor Entry form.

Troubleshooting the database

MarkView database objects provide the core framework for MarkView suite components. The following section describes how to administer database objects and the MarkView schema.

Check your current version

A standard installation includes the MV_Version object to show you the overall release number for that installation of the database objects.

1. Log in to your database as the MarkView schema owner.
2. At the SQL prompt, enter `set server output on size 100000;`

This command allocates space to display the version number for all components of the MarkView database objects. Once you specify the server output setting, the setting remains for the duration of your SQL*Plus session.

3. At the SQL prompt, enter `exec mv_version.showall`

   MarkView displays the version numbers for all database objects. The version number for the MV_Version object identifies the overall release number for the installed version of MarkView database objects.

   Packages ending with _CUSTOM may not report a version number.

Back up _CUSTOM packages

The _CUSTOM packages typically contain functionality and scripting that is unique to an installation of MarkView database objects. An upgrade uses the SQL CREATE OR REPLACE command on the objects it is upgrading. Because of this, you should back up any _CUSTOM packages at your site before upgrading MarkView database objects. Always store a copy of the SQL script files that create the current version of your _CUSTOM packages.

Before upgrading, always back up _CUSTOM packages. Failure to do so can result in the loss of your customizations.

Use the following procedure to extract the current version of your _CUSTOM packages from the database. If you overwrite a previous package without saving a copy, that version is lost.

1. Log in to SQL*Plus.
2. At the SQL prompt, enter `set pages xxx` where xxx is the number of lines per page. Specify a number greater than the number of lines in the _CUSTOM package body.
3. At the SQL prompt, enter `set lines xxx` where xxx is the number of characters per line. Specify a number greater than the number of characters per line in the _CUSTOM package body.
4. At the SQL prompt, enter `spool mv_package_name_custom_from_database.sql` to send all text to the specified spool file. If you are extracting multiple packages, give each one a unique filename.
5. At the SQL prompt, enter the following commands, where MV_PACKAGE_NAME_CUSTOM is the name of the _CUSTOM package to extract:

   ```
   select text
     from user_source
     where name = 'MV_PACKAGE_NAME_CUSTOM'
     and type = 'PACKAGE BODY'
     order by line;
   ```

   The command extracts the body of the package and saves data in the spool file.
6. At the SQL prompt, enter `add`.

   Output information no longer goes to the spool file.
7. Open the spool file in a text editor and add the following line before the package body:

   ```
   create or replace
   ```
8. Remove extraneous lines from the beginning and end of the file.
9. Save the file with the **.sql** file extension. Run the script to recreate the package body that you just extracted.

Add custom grants and synonyms

By default, the MarkView installer creates all the grants and synonyms required by the MarkView for Oracle Applications database objects. For companies with customizations, other MarkView components might require additional grants and synonyms. For example, workflow rules that access data in other Oracle Applications tables need grants and synonyms on these tables. Create custom grants and synonyms by adding entries to the object_data_custom.sql file and rerunning the grants and synonyms script.

object_data_custom file format

To create custom grants and synonyms, add new lines to the object_data_custom.sql file that is in the grants_synonyms directory under mvoadbobj. These lines are calls to the ObjData procedure. After you add these lines, rerun the grants and synonyms script to recognize the new settings.

ObjData syntax

ObjData ([Owner], [Object Name], [Synonym Name], [Ref By MarkView], [Ref By Support], [Ref By Gateway], [Ref By Apps], [Execute Priv], [Select Priv], [Insert Priv], [Update Priv], [Delete Priv], [With Grant Option], [Reference Type] [Can Be View]);

Where

Parameter	Description
[Owner]	The owner of the object: M—schema A—Oracle Applications schema
Object Name	The name of the object in the owner schema to receive grants and synonyms.
[Synonym Name]	The name of the synonym to create for the named object. An empty string or null value for this parameter directs the system to use the object name.
[Ref By...] parameters	Specifies that the schema requires access to the object: Y—requires access N—does not require access For example, Ref By MarkView=Y means the schema requires access to the object. Set Support and Gateway to N.
[...Priv] parameters	Indicates which privileges to grant to the schema on the object. Set these parameters to Y or N, as appropriate. Privileges can be Execute, Select, Insert, Update, and Delete.
[With Grant Option]	Y indicates that the grant option awards privileges on the object, which means that users with privileges on the object can grant those privileges to other users.

Parameter	Description
[Reference Type]	Indicates if grants only, synonyms only, or both grants and synonyms are to be created on the object. Set the parameter to G—creates grants only S—creates synonyms for accessing Oracle Applications multi-organization _ALL tables B—creates both grants and synonyms
[Can Be View]	Enables the creation of a view rather than a synonym on the object in the target schema. Set to Y or N. This parameter works in conjunction with the Replace Synonyms with Views option, which must be set to YES, for this parameter to be meaningful. For more information about advanced configuration options such as Replace Synonyms with Views, contact your system integrator.

Sample custom call

The installed version of object_data_custom.sql summarizes the call parameters to ObjData and includes examples, one of which appears below, with column headers shown for alignment. Set tabs every three characters to align parameters with column headers. End each procedure call with a semicolon (;).

The following example assumes that workflow rules in the MarkView schema require Select, Update, and Insert access to the AP_TABLE_ALL table in the Oracle Applications schema. Since this is a multi-organization table, two synonyms are created.

While the Apps schema may have a view AP_TABLE that displays the table for the current organization, the MarkView schema typically requires access to data from all organizations through a multi-organization table such as AP_TABLE_ALL. Thus the additional synonym (AP_TABLE) is created. The table looks similar to the following:

```
-- Object Object Synonym == Ref By ==Grant Privilege == Ref Can Be-- Owner Name == Name == MV Sup Gwy App
Exe Sel Ins Upd Del Opt Type View
ObjData ('A', 'AP_TABLE_ALL', '',          'Y', 'N','N','N','N','Y','Y','Y','N','N','B', 'N');
ObjData ('A', 'AP_TABLE_ALL', 'AP_TABLE', 'Y', 'N','N','N','N','N','N','N','N','N','S', 'N');
```

In the first call:

- The Apps schema (A) owns the object.
- The synonym name is an empty string that defaults to the object name (AP_TABLE_ALL).
- The reference type (B) creates both grants and synonyms.
- The MarkView schema (MV=Y) has access.
- Select, Insert, and Update privileges are granted (Y).

In the second call:

- The Apps schema (A) owns the object.
- The synonym name is AP_TABLE.
- Because the first call sets the necessary grants, the reference type (S) creates only synonyms.
- The MarkView schema (MV=Y) has access.
- Because no grants exist, no privileges are set.

Create the grants and synonyms

After adding custom grants and synonyms, rerun the script that creates all the grants and synonyms to include the newly defined objects.

To run the script, you need the following:

- Connection information (user name, password, and connect string) to both the MarkView and Apps schemas.
- The version of Oracle Applications on which you are running.
- A local copy of the MarkView for Oracle Applications database objects in a directory to which you have write permission.
- The object_data_custom file that you updated, in place of the originally installed version.

When you create grants and synonyms, the calls in object_data_custom.sql are validated, and errors are output to the log file. Most errors are trapped prior to creating grants and synonyms (both standard and custom) except when the object name is invalid or does not exist in the specified schema. This type of error can only be trapped while attempting to create the grants.

Examine the mvoa_create_grants_synonyms.log file in the grants_synonyms directory for errors. As needed, rerun the script with a logging level of verbose to obtain more details about the errors in question. Also try using Log Only mode, another advanced configuration option.

1. Log in to SQL*Plus and set the working directory to **mvoadbobj**.
2. At the SQL prompt, run the following script:

   ```
   @grants_synonyms/mvoa_create_grants_synonyms.sql
   ```
3. Answer the prompts as the script executes.
4. When the script finishes processing, recompile database objects as necessary.

Check rule execution time

Periodically monitor the rule execution time. A rule that takes much longer than average to execute serves as a potential warning of database problems.

From a PL/SQL prompt, run the following series of SQL and SQL*Plus statements:

```
set verify off
spool perfstat.log
accept sdate prompt 'Enter the start date (../YYYY): '
accept edate prompt 'Enter the end date (../YYYY): '
set pagesize 500
column "Avg Time (sec)" format 999999.999999
select rt.rule_type_name "Rule Type Name",
       avg(tr.processing_time) "Avg Time (sec)",
       max(tr.processing_time) "Max Time (sec)",
       count(*) "Executions"
from   sf_rule_type        rt,
       sf_rule_instance    ri,
       sf_triggered_rule   tr,
       sf_event_occurrence eo
where  tr.rule_instance_id = ri.rule_instance_id
and    ri.rule_type_id = rt.rule_type_id
and    tr.event_occurrence_id = eo.event_occurrence_id
and    eo.processed_timestamp between to_date('&&sdate', '../YYYY') and
```

```
       to_date('&&edate', '../YYYY')
and    tr.processing_time is not null
group  by rt.rule_type_name
order  by 2 desc
/
spool off
```

These statements return and log the average rule execution time as well as the maximum rule execution time. Save your log files to maintain a performance history for your database.

Save this set of statements to rerun later. For example, if you save the set as perfstat.sql, you can run it later using the command @perfstat. The results are saved to a file called perfstat.log.

Synchronize password changes

Many corporate IT policies require periodic password changes for the database schemas and administrator accounts. Implementing such a policy with the MarkView schema and MarkView Administrator account passwords impact many MarkView suite components. Propagate the password changes across the following components:

- All MarkView Capture and Output components.
- All application servers where a MarkView data source is defined.

Update the MarkView schema password on the WebLogic application server

Update your Connection Pool in the WebLogic Console.

1. Log in to the WebLogic Server Administration Console and navigate to **Domain Structure** > **Services** > **Datasources**.
2. Select the **MarkViewDS** data source and click the **Connection Pool** tab.
3. Update the Password and Confirm Password fields.

Update the MarkView schema password on the WildFly application server

Update your Connection Pool in the WildFly Console.

1. Log in to the WildFly Administration Console and navigate to **Configuration** > **Datasources**.
2. Select the **MarkViewDS** data source and click the **Security** tab.
3. Click **Edit** and update the Password and Confirm Password fields.

Update Capture and Output component passwords

After changing the MarkView schema password

If you changed the MarkView Schema password, complete the procedures in this section.

Update the MarkView Import Server password

1. Stop the **MarkView Import Server** service.

2. Launch the **MarkView Import Server - Configuration** tool.
3. Select the import server you are using and click **Preferences**.
4. In the Database Password field, enter the new password.
5. Click **Test DB** to verify your entry.
6. Click **OK**.

Update the MarkView Data Export Server password

1. Stop the **MarkView Data Export Server** service.
2. Launch the **MarkView Data Export Server - Configuration** tool.
3. Select the export server you are using and click **Preferences**.
4. In the Database Password field, enter the new password.
5. Click **Test DB** to verify your entry.
6. Click **OK**.

Update the MarkView Barcode Server password

1. Stop the **Barcode Server** service.
2. Open the **MarkView Barcode Server - Configuration** tool.
3. Select the instance that you are running and click **Preferences**.
4. In the DbPassword field, enter the new password.
5. Click **Test** to verify your entry.
6. Click **OK**.

Update the MarkView Mail Gateway password

1. Stop the **Mail Gateway** service.
2. Open the **MarkView Mail Gateway - Configuration** tool.
3. Select the instance that you are running and click **Preferences**.
4. In the Database Password field, enter the new password.
5. Click **Test DB** to verify your entry.
6. Click **OK**.

Update theKofax MarkView Export Connector password

1. Run the **MVImportAPIConfig.exe** file.
2. Select the **MarkView Import API Configuration** and click **Preferences**.
3. In the Database Password field, enter the new password.
4. Click **Test DB** to verify your entry.
5. Click **OK**.

Update the Kofax Transformation Modules password

1. On the Capture and Output server, on the Start menu, select **Programs** > **Kofax Transformation** > **Project Builder**.
2. Open the project that is synchronized with your batch.
3. On the menu bar, select **Project** > **Script Variables**.

4. Update the DatabasePWD script variable.
5. Click **OK**.

Update the Relational Database passwords

If your project settings in Kofax Capture include any relational databases, update those passwords as follows:

1. In Kofax Capture, right-click the project name and select **Project Settings**.
2. Click the **Databases** tab.
3. Select a relational database line, for example, LimLockDB, and click **Properties**.
   See the Database Type column to identify relational databases.
4. Next to the Connection String field, click **Configure**.
5. In the dialog box that opens, update Password.
6. Click **Test Connection**.
7. Click **OK**.

> After updating passwords in Kofax Capture and Kofax Transformation Modules, synchronize and publish the project in Kofax Capture.

Update the Database Validation Properties in Kofax Capture batch classes

1. In Kofax Capture Administration, expand each Batch Class.
2. Right-click the Document Type and select **Database Validation**.
3. Click **Properties**.
4. In the Password field, enter the new password.
5. Click **OK**.

Request denied due to unacceptable characters

If you attempt to enter unsupported characters or words in MarkView entry fields, the following message appears:

```
Request has been denied due to unacceptable characters in the input.
```

You can resolve the issue by removing the following unsupported characters or words from the entry.

Unsupported Characters/Words	Description
<	Less than symbol
>	Greater than symbol
"	Opening or closing quotation mark
#	Number sign
background:url	Reserved word

Unsupported Characters/Words	Description
javascript	Reserved word

Reset SAML preferences

If SAML preferences are set to incorrect values, the MarkView applications might fail to start.

To reset the SAML preferences:

1. Log in to SQL*Plus using the MarkView schema user name and password.
2. From the PL/SQL prompt, run the following script:

```
declare
  procedure ResetSamlPreference(Preference varchar2) is
  begin
    MV_Preference_Util.DeleteSystemPreference(Preference);
    exception when others then null;
  end ResetSamlPreference;
begin
  ResetSamlPreference('AUTH_SSO_SAML_IDP_METADATA_PATH');
  ResetSamlPreference('AUTH_SSO_SAML_KEYSTORE_PATH');
  ResetSamlPreference('AUTH_SSO_SAML_KEYSTORE_PASSWD');
  ResetSamlPreference('AUTH_SSO_SAML_CERT_NAME');
  ResetSamlPreference('AUTH_SSO_SAML_CERT_PASSWD');
  commit;
end;
/
```

Appendix B

Oracle configuration parameters

Enable Tax Classification Codes

To use the Release 11i model (Tax Classification Codes) in Oracle EBS R12 or 12.2, generate a new ACD configuration using Use11iTaxMode = TRUE.

By default, the ACD uses the standard Regime to Rate Flow.

Example:

```
declare
  dfm_definition_data MVERP_DFM_Util.DFMDefinitionData_R;
  dfm_fields          MVERP_DFM_Generate.GenerateFields_T;

begin
  -- extract current DFM data
  MVERP_DFM_Generate.GetToolDfmDefinition(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        dfmfields => dfm_fields);

  -- set new values
  dfm_definition_data.Use11iTaxMode := TRUE;

  -- save DFM data
  MVERP_DFM_Generate.GenerateExpressMarkup(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        DFMfields => dfm_fields);

end;
/
```

Add the Automatic Distributions check box

Configure Accounting Details to display a check box that instructs MarkView to automatically create distributions based on the accounting line data being entered.

To display the check box, generate a new ACD configuration using ShowAutoGenerateDistOption = TRUE.

By default, this check box does not appear.

Example:

```
declare
  dfm_definition_data MVERP_DFM_Util.DFMDefinitionData_R;
  dfm_fields          MVERP_DFM_Generate.GenerateFields_T;

begin
  -- extract current DFM data
  MVERP_DFM_Generate.GetToolDfmDefinition(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        dfmfields => dfm_fields);

  -- set new values
  dfm_definition_data.ShowAutoGenerateDistOption := TRUE;

  -- save DFM data
  MVERP_DFM_Generate.GenerateExpressMarkup(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        DFMfields => dfm_fields);

end;
/
```

Set default values for the Automatic Distributions check box

The Accounting Details (ACD) tool in MarkView lets you supply coding information for invoices during the invoice life cycle. Oracle R12 introduced the concept of invoice lines and invoice distributions. A line can have multiple distributions. However, many customers maintain a 1:1 relationship between invoice lines and distributions. MarkView provides a check box on the window that you can enable when adding lines to create a single distribution for the line automatically. By default, this check box is disabled.

MarkView includes a preference that lets you change the default setting (disabled) by updating the definition of the ACD tool.

To control the default status of the check box, set ShowAutoGenerateDistOption to **TRUE** and **AutoGenerateDistDefValue** to one of the following:

- TRUE: Enables the check box by default
- FALSE: Disables the check box by default
- NULL: Disables the check box by default

Example:

```
declare
  dfm_definition_data MVERP_DFM_Util.DFMDefinitionData_R;
  dfm_fields          MVERP_DFM_Generate.GenerateFields_T;

begin
  -- extract current DFM data
   MVERP_DFM_Generate.GetToolDfmDefinition(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        DFMFields => dfm_fields);

  -- set new values
  dfm_definition_data.ShowAutoGenerateDistOption := TRUE;
```

```
  dfm_definition_data.AutoGenerateDistDefValue := TRUE;

  -- save DFM data
  MVERP_DFM_Generate.GenerateExpressMarkup(
        ToolName   => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        DFMFields         => dfm_fields);

end;
/
```

Set default values for Tax fields

Use the Accounting Details (ACD) tool in MarkView to process the Line Type selection and to default the Tax fields. If you enable the Line Type Defaults option, the SetLineTypeDefaultsForLine or SetLineTypeDefaults custom callouts return the default values for the Tax fields after you select Line Type in the Accounting Details form. For more information, see MVERP_DFM_CUSTOM on page 34.

Example:

```
declare
  dfm_definition_data MVERP_DFM_Util.DFMDefinitionData_R;
  dfm_fields          MVERP_DFM_Generate.GenerateFields_T;

begin
  -- extract current DFM data
  MVERP_DFM_Generate.GetToolDfmDefinition(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        dfmfields => dfm_fields);

  -- set new values
  dfm_definition_data.EnableLineTypeDefaults := TRUE;

  -- save DFM data
  MVERP_DFM_Generate.GenerateExpressMarkup(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        DFMfields => dfm_fields);

end;
/
```

Set maximum values for frequently used accounts

The account information in MarkView Viewer is retrieved from the ERP system, which may be time-consuming. Use the FrequentAccountsLOVSize and FrequentAccountsHistorySize parameters to set the maximum length of the list of values to be displayed in MarkView Viewer and the maximum number of accounts that a user can save in the database. By default, both parameters are set to 50.

Example:

```
declare
    dfm_definition_data MVERP_DFM_Util.DFMDefinitionData_R;
    dfm_fields MVERP_DFM_Generate.GenerateFields_T;
```

```
begin
    -- extract current DFM data
    MVERP_DFM_Generate.GetToolDfmDefinition(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        dfmfields => dfm_fields);

    -- set new values
    dfm_definition_data.FrequentAccountsLOVSize := 100;
    dfm_definition_data.FrequentAccountsHistorySize := 150;

    -- save DFM data
    MVERP_DFM_Generate.GenerateExpressMarkup(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        DFMfields => dfm_fields);
end;
/
```

Enable descriptive flex fields

Generate a new ACD configuration to enable descriptive flex fields.

Example:

```
declare
  dfm_definition_data mverp_dfm_util.DFMDefinitionData_R;
  dfm_fields mverp_dfm_generate.GenerateFields_T;
begin
  -- extract current DFM data
  mverp_dfm_generate.GetToolDfmDefinition(
  ToolName => 'Distributions',
  DFMDefinitionData => dfm_definition_data,
  dfmfields => dfm_fields);
  -- set new values
  for i in nvl(dfm_fields.first, 0)..nvl(dfm_fields.last, -1)
  loop
    if dfm_fields(i).FieldName = MVERP_DFM_Util.DFM_FIELD_DFF then
      dfm_fields(i).DisplayFieldYN := 'Y';
      dfm_fields(i).SummaryFieldYN := 'Y';
      dfm_fields(i).ModifyFieldYN  := 'Y';
    end if;
  end loop;
  -- save DFM data
  mverp_dfm_generate.GenerateExpressMarkup(
  ToolName => 'Distributions',
  DFMDefinitionData => dfm_definition_data,
  dfmfields => dfm_fields);
end;
/
```

Enable project fields

If the project fields are enabled in Oracle EBS, you must enable these fields in MarkView to edit lines.

Generate a new ACD configuration to enable the project fields, such as project, task, expenditure type, expenditure org, and date of expenditure.

Example:

```
declare
  dfm_definition_data mverp_dfm_util.DFMDefinitionData_R;
  dfm_fields mverp_dfm_generate.GenerateFields_T;
begin
  -- extract current DFM data
  mverp_dfm_generate.GetToolDfmDefinition(
  ToolName => 'Distributions',
  DFMDefinitionData => dfm_definition_data,
  dfmfields => dfm_fields);
  -- set new values
  for i in nvl(dfm_fields.first, 0)..nvl(dfm_fields.last, -1)
  loop
    if dfm_fields(i).FieldName in (
    MVERP_DFM_Util.DFM_FIELD_PROJECT,
    MVERP_DFM_Util.DFM_FIELD_TASK,
    MVERP_DFM_Util.DFM_FIELD_EXP_TYPE,
    MVERP_DFM_Util.DFM_FIELD_EXP_ORG,
    MVERP_DFM_Util.DFM_FIELD_EXP_ITEM_DATE
    ) then
      dfm_fields(i).DisplayFieldYN := 'Y';
      dfm_fields(i).SummaryFieldYN := 'Y';
    end if;
  end loop;
  -- save DFM data
  mverp_dfm_generate.GenerateExpressMarkup(
  ToolName => 'Distributions',
  DFMDefinitionData => dfm_definition_data,
  dfmfields => dfm_fields);
end;
/
```

Enable Income Tax

To enable the Income Tax Region field, you must also enable the Income Tax Type field.

To enable editing for the Income Tax Region and Income Tax Type fields in MarkView Viewer Accounting and to display these fields in Additional Details, generate a new ACD configuration.

Example:

```
declare
  dfm_definition_data mverp_dfm_util.DFMDefinitionData_R;
  dfm_fields mverp_dfm_generate.GenerateFields_T;
begin
  -- extract current DFM data
  mverp_dfm_generate.GetToolDfmDefinition(
  ToolName => 'Distributions',
  DFMDefinitionData => dfm_definition_data,
  dfmfields => dfm_fields);
  -- set new values
  for i in nvl(dfm_fields.first, 0)..nvl(dfm_fields.last, -1)
  loop
    if dfm_fields(i).FieldName in (
    MVERP_DFM_Util.DFM_FIELD_TAX_REGION,
```

```
    MVERP_DFM_Util.DFM_FIELD_TYPE_1099
    ) then
      dfm_fields(i).DisplayFieldYN := 'Y';
      dfm_fields(i).SummaryFieldYN := 'Y';
    end if;
  end loop;
  -- save DFM data
  mverp_dfm_generate.GenerateExpressMarkup(
  ToolName => 'Distributions',
  DFMDefinitionData => dfm_definition_data,
  dfmfields => dfm_fields);
end;
/
```

Add Undistributed Amount

To display Undistributed Amount in the Accounting detail section and in Additional Details for invoice lines, generate a new ACD configuration with the ShowDistributionTotal parameter set to TRUE. By default, ShowDistributionTotal is TRUE.

To hide Undistributed Amount, generate a new ACD configuration with the ShowDistributionTotal parameter set to FALSE.

Example:

```
declare
  dfm_definition_data MVERP_DFM_Util.DFMDefinitionData_R;
  dfm_fields          MVERP_DFM_Generate.GenerateFields_T;
begin
  -- extract current DFM data
  MVERP_DFM_Generate.GetToolDfmDefinition(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        dfmfields => dfm_fields);
  -- set new values
  dfm_definition_data.ShowDistributionTotal := TRUE;
  -- save DFM data
  MVERP_DFM_Generate.GenerateExpressMarkup(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        DFMfields => dfm_fields);
end;
/
```

Use tax default hierarchy

To use the tax default hierarchy, you must enable tax classification codes.

If you enable the UseTaxDefaultHierarchy parameter, the default Tax Classification value is automatically set for the Invoice line in MarkView Viewer Accounting.

To use the default Tax Classification value, generate a new ACD configuration with the UseTaxDefaultHierarchy parameter set to TRUE. By default, UseTaxDefaultHierarchy is FALSE.

Example:

```
declare
  dfm_definition_data MVERP_DFM_Util.DFMDefinitionData_R;
  dfm_fields          MVERP_DFM_Generate.GenerateFields_T;
begin
  -- extract current DFM data
  MVERP_DFM_Generate.GetToolDfmDefinition(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        dfmfields => dfm_fields);
  -- set new values
  dfm_definition_data.Use11iTaxMode := TRUE;
  dfm_definition_data.UseTaxDefaultHierarchy := TRUE;
  -- save DFM data
  MVERP_DFM_Generate.GenerateExpressMarkup(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        DFMfields => dfm_fields);
end;
/
```

Allow post validation changes

To edit distribution lines that are already validated but not yet approved, generate a new ACD configuration with the AllowPostValidationChanges parameter set to TRUE. By default, AllowPostValidationChanges is FALSE.

Example:

```
declare
  dfm_definition_data MVERP_DFM_Util.DFMDefinitionData_R;
  dfm_fields          MVERP_DFM_Generate.GenerateFields_T;
begin
  -- extract current DFM data
  MVERP_DFM_Generate.GetToolDfmDefinition(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        dfmfields => dfm_fields);
  -- set new values
  dfm_definition_data.AllowPostValidationChanges := TRUE;
  -- save DFM data
  MVERP_DFM_Generate.GenerateExpressMarkup(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        DFMfields => dfm_fields);
end;
/
```

Enable posted distribution lines changes

Before you configure this parameter, you must set AllowPostValidationChanges to TRUE.

To edit distribution lines that are already validated and posted in the general ledger, generate a new ACD configuration with the EnablePostedLineChanges parameter set to TRUE. By default, EnablePostedLineChanges is FALSE.

Example:

```
declare
  dfm_definition_data MVERP_DFM_Util.DFMDefinitionData_R;
  dfm_fields          MVERP_DFM_Generate.GenerateFields_T;
begin
  -- extract current DFM data
  MVERP_DFM_Generate.GetToolDfmDefinition(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        dfmfields => dfm_fields);
  -- set new values
  dfm_definition_data.AllowPostValidationChanges := TRUE;
  dfm_definition_data.EnablePostedLineChanges := TRUE;
  -- save DFM data
  MVERP_DFM_Generate.GenerateExpressMarkup(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        DFMfields => dfm_fields);
end;
/
```

Use withholding tax default hierarchy

In MarkView Viewer, you cannot add or edit withholding taxes for invoice lines and distributions.

If UseWHTDefaultHierarchy is TRUE, the withholding tax and the payment withholding tax values are automatically populated as specified for the invoice header. The distribution line withholding tax values are automatically populated as specified for the invoice line.

To use the withholding default tax hierarchy, generate a new ACD configuration with the UseWHTDefaultHierarchy parameter set to TRUE. By default, UseWHTDefaultHierarchy is FALSE.

Example:

```
declare
  dfm_definition_data MVERP_DFM_Util.DFMDefinitionData_R;
  dfm_fields          MVERP_DFM_Generate.GenerateFields_T;
begin
  -- extract current DFM data
  MVERP_DFM_Generate.GetToolDfmDefinition(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        dfmfields => dfm_fields);
  -- set new values
  dfm_definition_data.UseWHTDefaultHierarchy := TRUE;
  -- save DFM data
  MVERP_DFM_Generate.GenerateExpressMarkup(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        DFMfields => dfm_fields);
```

```
end;
/
```

Custom account validation

If you need to disable the use of the custom account validation, generate a new ACD configuration with the UseCustomAccountValidation parameter set to FALSE. By default, UseCustomAccountValidation is TRUE.

Example:

```
declare
  dfm_definition_data MVERP_DFM_Util.DFMDefinitionData_R;
  dfm_fields          MVERP_DFM_Generate.GenerateFields_T;
begin
  -- extract current DFM data
  MVERP_DFM_Generate.GetToolDfmDefinition(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        dfmfields => dfm_fields);
  -- set new values
  dfm_definition_data.UseCustomAccountValidation := TRUE;
  -- save DFM data
  MVERP_DFM_Generate.GenerateExpressMarkup(
        ToolName => 'Distributions',
        DFMDefinitionData => dfm_definition_data,
        DFMfields => dfm_fields);
end;
/
```

Appendix C

Third-party license agreement

BEA Public License Version 2.1

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.
   - "License" shall mean the terms and conditions of this agreement.
   - "Licensor" shall mean BEA Systems, Inc.
   - "Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means

     (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or

     (ii) ownership of fifty percent (50%) or more of the outstanding shares, or

     (iii) beneficial ownership of such entity.
   - "You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License, including but not limited to each Contributor other than Licensor in such Contributor's role as a licensee for the Software.
   - "Source Format" shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.
   - "Object Format" shall mean any form resulting from mechanical transformation or translation of a Source Format, including but not limited to compiled object code, generated documentation, and conversions to other media types.
   - "Software" shall mean the original version of the software accompanying this agreement as released by BEA, including in Source or Object Format, and also any documentation provided therewith.
   - "Derivative Works" shall mean any work, whether in Source or Object Format, that is based on (or derived from) the Software and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Software and derivative works thereof.
   - "Contribution" shall mean any work of authorship, including the original version of the Software and any modifications or additions to that Software or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Software by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of

discussing and improving the Software, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

- "Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Software.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Software and such Derivative Works in Source or Object Format. Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the foregoing copyright license.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Software, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Software to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Software or a Contribution incorporated within the Software constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Software shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Software or Derivative Works thereof in any medium, with or without modifications, and in Source or Object Format, provided that You meet the following conditions:

- You must give any other recipients of the Software or Derivative Works a copy of this License; and
- You must cause any modified files to carry prominent notices stating that You changed the files; and
- You must retain, in the Source Format of any Derivative Works that You distribute, BEA's copyright notice, "© [Date] BEA Systems, Inc. All rights Reserved.", and all other copyright, patent, trademark, and attribution notices from the Source Format of the Software, excluding those notices that do not pertain to any part of the Derivative Works; and
- You must affix to the Software or any Derivative Works in a prominent manner BEA's copyright notice, "(© [Date] BEA Systems, Inc. All rights Reserved." whenever You distribute the Software or such Derivative Works in Object Format.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Software otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Software by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Software and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using and distributing the Software and assume all risks associated with Your exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations. Further, You understand that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that its Contribution does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to You for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, You hereby assume sole responsibility to secure any other intellectual property rights needed, if any.
8. Limitation of Liability. EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NO CONTRIBUTOR SHALL HAVE ANY LIABILITY TO YOU FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE SOFTWARE OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
9. Accepting Warranty or Additional Liability. Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Software, if You include the Software in a commercial product offering, You may do so only in a manner which does not create potential liability for any Contributor. Therefore, if You include the Software in a commercial product offering, You shall and hereby do agree to defend and indemnify each and every Contributor against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against such Contributor(s) to the extent caused by Your acts or omissions in connection with Your distribution of the Software in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify to receive indemnification from You, a Contributor must:
    a. promptly notify You in writing of such claim, and
    b. allow the You to control, and cooperate with the You in, the defense and any related settlement negotiations.

    The Contributor indemnified by You may participate in any such claim at its own expense.