METALOGIX TRANSFORMERS REFERENCE

Abstract

Introduce the Metalogix Transformations framework, and how to use it. A reference to the objects available when running the Metalogix "Invoke PowerShell Script" transformer type.

Metalogix Transformers Reference

Contents

Introduction

This guide introduces the Metalogix Transformations framework, and how to use it.

Prerequisites

Prior to reading this guide, a base understanding of PowerShell is advantageous. The Metalogix PowerShell Reference provides an introduction to many of the basic PowerShell concepts used in this document.

What are Metalogix Transformations

The Metalogix Transformations framework is a set of hooks into the Metalogix actions that allow programmatically minded end users to view or transform the way that object data lives at that point in the action. The term transformer refers to a specific instance of a transformation.

Tip: Almost all transformers are created by inserting PowerShell scripts into the action hooks, so understanding PowerShell will provide a strong foundation for writing transformers.

When to use Transformations

Transformations are written in order to solve a customer requirement that Content Matrix does not solve out of the box. Due to the required familiarity with the Metalogix object model and high degree of difficulty involved in writing transformations, they are very rarely written by customers themselves. In some situations, partners may be engaged to write transformers for customers. In these situations, please consult your partner / alliance manager for a list of qualified partners.

Use Transformations to:

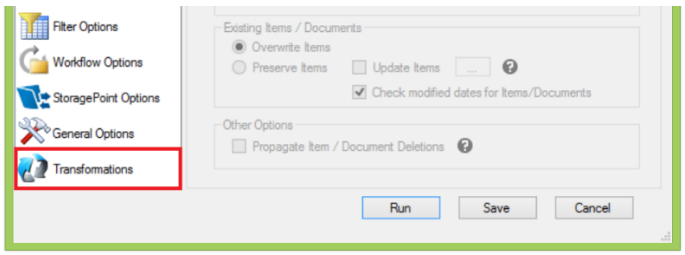
- Modifying properties and settings on SharePoint objects during a migration
- Applying new information onto objects during a migration
- Filtering objects from migration based on a migration condition

Don't use Transformations to:

- Change fundamental object types (e.g. site to list, item to document, etc)
- Perform lengthy calculations. An improperly written Transformer can slow a migration to a crawl

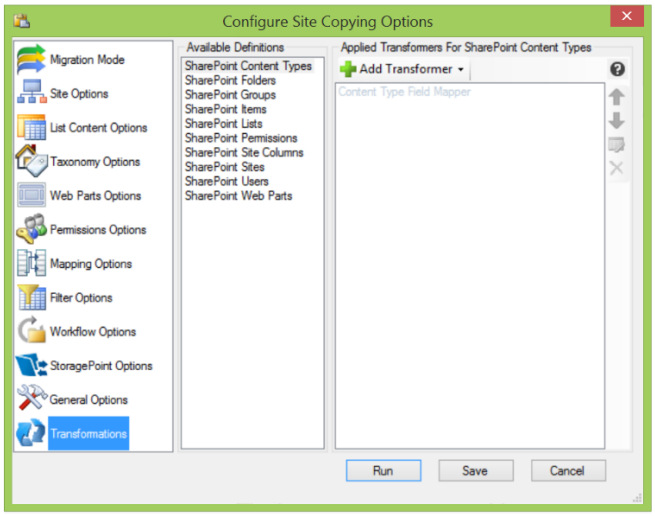
Creating Transformations

Most migration actions in Content Matrix have the ability to use transformations. If the desired action has hooks for transformations placed within it, the configuration dialog will include a left navigation option labeled "Transformations":



Viewing Transformers

Clicking the option labeled “Transformations” will load the transformations control:

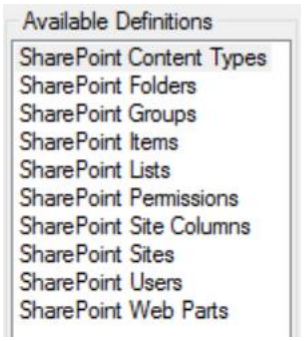


The control is separated into two sections:

1) Available Definitions
2) Applied Transformers For SharePoint <definition type>

Available Definitions

This section of the control lists different object types that support transformers in this action configuration. For example, when configuring a site copy action you have the following object types that support the creation of transformers:



Choosing a definition will allow the creation and application of transformers for that definition, and enables the next section of the control.

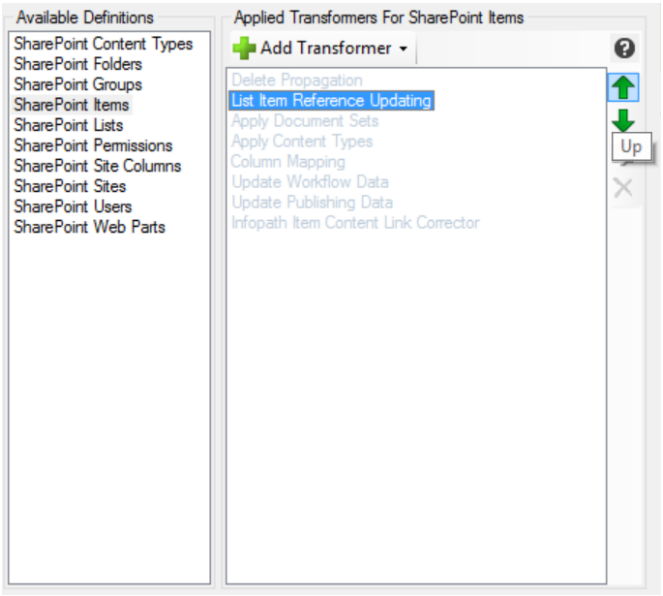
Tip: The definition of a transformer will change the objects referenced within it. If you choose the Lists definition, then the object the transformer will make available to scripts will be a SharePoint List. Generally you will choose the type of object that you wish to transform as the definition.

Applied Transformers For SharePoint <definition type>

This section of the control is dependent on which definition type is chosen in the previous section, and allows the creation, application and reordering of transformers.

Once a definition type is chosen in the “Available Definitions” control, this section will list all the applied transformers for the chosen definition type.

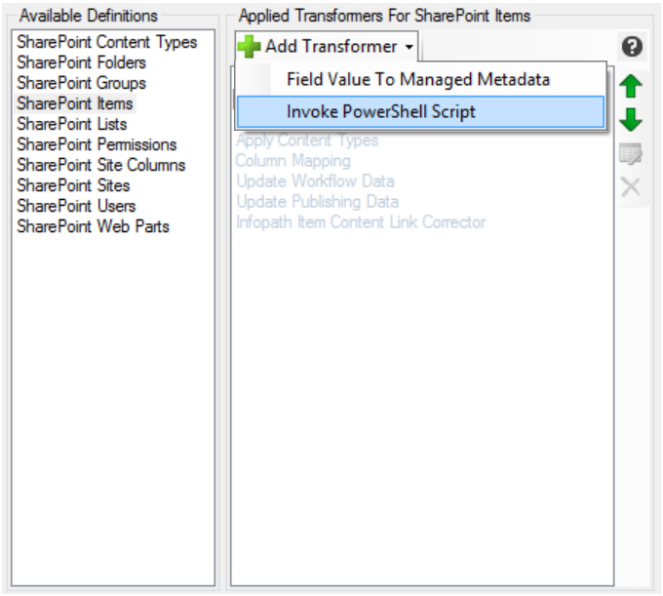
Default transformers will also be listed in this control, and are identified by being greyed out in the UI.



Tip: Default transformers are required, and thus cannot be edited or deleted, but the order in which they run can be changed by selecting the desired transformer and clicking the green arrow buttons on the right hand side of the control to move it up or down in the order.

Adding Transformers

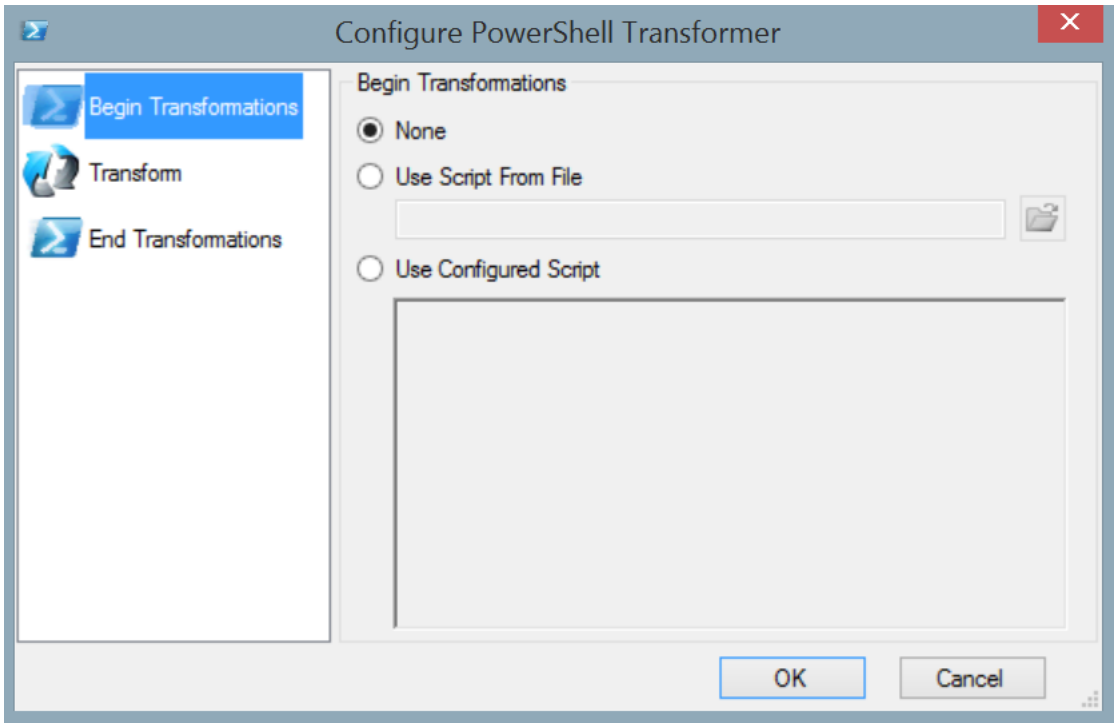
Once an object definition has been chosen, new transformers can be added through the “Add Transformer” dropdown in the available transformers control for that definition:



Tip: The dropdown will contain all available transformer types for that object definition. It is not currently possible to write additional transformer types, but the "Invoke PowerShell Script" type is always available.

Configuring the "Invoke PowerShell Script" transformer

After clicking the "Invoke PowerShell Script" transformer type, the configuration dialog will open:



There are three (3) primary sections of the configuration dialog:

1) Begin Transformations
2) Transform
3) End Transformations

Begin Transformations

The begin transformations step runs when the collection of objects to be migrated is loaded, and allows the user to initialize objects and make changes to objects that will be reflected through the migration of the entire collection.

Tip: Objects created in the begin transformations step are available throughout the remainder of the collection copy, so try using it to initialize collections and variables that you'll need to use multiple times. A good example of this is to load an excel file into memory and extract values into variables in the begin step instead of during the individual item transformations.

The begin transformations step has visibility into the collection of source objects, target objects, and the action being run, and surfaces each as a variable listed below.

Object variables

- $targets

    - The collection of target objects. Can be used as a window to other target objects, like the target parent list or site.
- $sources
    - The collection of source objects. Can be used as a window to other source objects, like the source parent list or site.
- $action
    - The operation being run. Contains information about the current status of the operation.

Tip: The object variable sections in this document are provided as an immediate window into the transformation objects available during different steps of the process, but the explanation in this document is limited. Try referring to the Transformer Object Model document provided with this guide for a more in-depth explanation.

Transform

The transform step runs once for each item in the collection of objects to be migrated. Due to the individual scope of this step, it has visibility into each object as well as the collection of sources and targets available in the other steps.

This visibility into individual objects is how the transformation of data occurs, and is surfaced through the special $dataObject variable.

Tip: The $dataObject variable will always be of the Metalogix object model type of the previously chosen transformer definition. For example, if the SharePoint List Items definition is chosen, the $dataObject variable will be of type (Metalogix OM) SPListItem, and will contain all metadata fields and methods that the SPListItem object contains.

Object variables

- $dataObject
    - The special variable only available in the transform step. This variable stores the actual object being migrated by the Metalogix action, which can be transformed by manipulating its properties and methods.
- $targets
    - See above
- $sources
    - See above
- $operation
    - See above

End Transformations

The end transformations step is run during the collection cleanup at the end of the copy. This step should be used to clean up and dispose any remaining variables or objects created during the transformations.

Tip: This step has visibility into target objects created during the collection migration. Try accessing target objects to see what they look like after the copy.

Object variables

The end transformations step has the same variable visibility as the begin transformations step.

Specifying the script

Each transformation step has a corresponding control that allows the configuring user the ability to specify how the PowerShell transformer will be run. There are three (3) options:

1) None
2) Use Script From File
3) Use Configured Script

None

This option is selected by default for each step and specifies that for this step, no action will be taken.

Use Script From File

This option allows the user to specify a path to a pre-created PowerShell script file to be used during the copy. The file must be available to the application at the time of migration.

Use Configured Script

This option allows the user to write PowerShell scripts directly into the migration configuration window.

Transformer Example 1

In this section, we will walk through the creation of a simple transformer based off of the previously defined steps.

Tip: In this example PowerShell tools are used that have not been discussed in this document, including conditional statements. Please review the PowerShell guide for a better understanding of how to use these tools.

Customer Use Case

All items within document libraries must be flattened to the root of the document library during a site migration.

Write the PowerShell script

Define requirements:

R1) Move items from within folders in document libraries to the root of the document library.

Define objects that must be modified from requirements:

1) Consulting the items view in Content Matrix indicates that SharePoint list items store their location within folders in a metadata field called FileDirRef. It is a string field. Modifying this field to the empty string manually confirms that items flatten to the root when it is modified.

   Design: FileDirRef must be transformed to the empty string mid migration to flatten items.

Define logical conditions:

1) The transformer must only execute on items within document libraries.

   Design: A conditional if block must determine whether the parent list of the item is a document library, skipping the transformation logic if it is not.

2) SharePoint folders are also list items, and removing the FileDirRef field on folders causes errors, so the transformer must not act on folders when they are encountered.

Design: A conditional if block must determine whether an item is a folder, and skip execution if it is.

Define error handling:

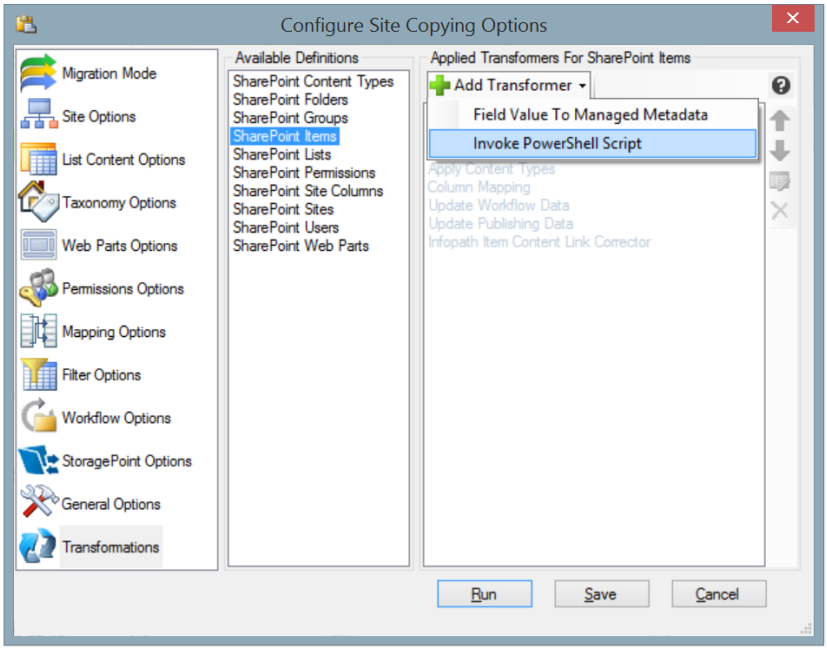
a. The Metalogix transformer exception catching will provide sufficient information if the transformer fails.
    i. **Design:** No additional error handling required.

Write the script in PowerShell ISE

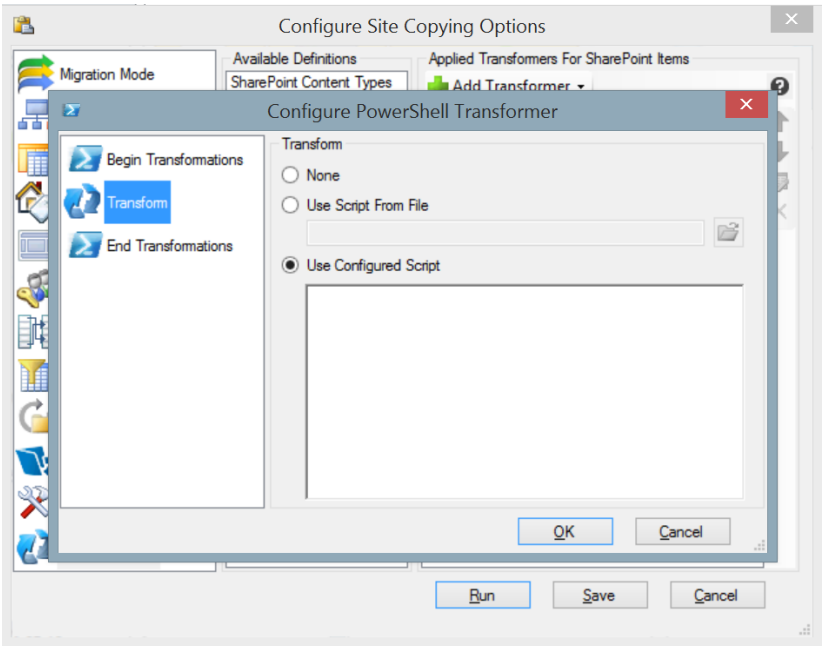
```
# Check to see if the parent list is a document library, then if the item is a folder
if($dataObject.ParentList.IsDocumentLibrary -and $dataObject["FSObjType"] -ne "1")
{
    $dataObject.FileDirRef = ""
}
```

Configure the action

1) The use case specifies the migration of a SharePoint site, so we will first configure the desired settings for our site copy. When we have completed the configuration for copying a site, we will choose the transformations option at the bottom of the left selection tab.
2) We will then choose the "SharePoint Items" definition from the "Available Definitions" control, and click the "Add transformer" dropdown, choosing the "Invoke PowerShell Script" type:



3) In the resulting configuration dialog, we will click the "Transform" step, and click the "Use Configured Script" radio button:



4) Paste the script previously created into the "Use Configured Script" window, and click OK.
5) Save, or run the action.

Transformer Results

Running a transformer can have multiple results:

1) Failure with an error
2) Migration succeeds, but desired effect not achieved
3) Migration succeeds, and desired effect achieved

Tip: It is very important to verify on the target whether a transformer has succeeded or not, as transformers do not log to the job listing unless they fail with an exception.

Failure with an error

Transformers can fail with errors for myriad reasons. In this case, it is important to double click the error message in the log, and read the exception text. In most situations, the error text reflects the actual error in the transformer script.

Migration succeeds but desired effect not achieved

This situation indicates that the script was written in a way that passes PowerShell execution syntax, but doesn't transform the object correctly.

Tip: A common mistake is to use incorrect syntax when attempting to access SPListItem fields. Some fields on a Metalogix SPListItem are exposed as properties on the object, and can be accessed by the property getter, but others are internal to the XML definition only, and must be accessed by a direct index query. Attempting to access a field that doesn't have a property association will not throw an exception, but it won't modify the property. Example: Using $dataObject.CustomField fails, but $dataObject["CustomField"] works.

Migration succeeds, with desired effect

This is the desired outcome. With this outcome, it is important to ensure that all possible cases have been accounted for with logical flow controls, and the script ensures that no matter the execution path, a successful migration is achieved.

Transformer Definitions

Tip: Many of the following definitions have an XML property that can be used to examine the full set of properties for the object. It is often useful to print the XML of the object to a file that can be examined and manipulated.

Content Types

Content type transformers execute during a content type collection copy.

Variables

- $dataObject
    - Type: SPContentType
    - Notable Members:
        - Property: Name
            - The name of the content type
        - Property: ContentTypeID
            - The ID of the content type
        - Property: ContentTypeXML
            - The full XML of the content type
- $sources / $targets
    - Notable members:
        - ParentList
            - The parent list of the content type collection on the respective server
        - ParentWeb
            - The parent web of the content type collection on the respective server

Folders

Folder transformers execute during a specific folder copy (in any other copy, folders are included as part of the list item copy)

Variables

- $dataObject
    - Type: SPFolder
    - Notable Members:
        - Property: Name
            - The name of the folder
        - Property: XML
            - The full XML of the folder
        - Property: ParentList
            - The parent list of the folder
        - Property: ServerRelativeURL
            - The full URL of the folder
- $sources / $targets
    - Notable members:
        - ParentList
            - The parent list of the folder set on the respective server
        - ParentWeb

- The parent web of the folder set on the respective server

Groups

Group transformers execute during a referenced groups and users copy

Variables

- $dataObject
  - Type: SPGroup
  - Notable members:
    - Property: Name
      - The name of the group
    - Property: Owner
      - The owner of the group
    - Property: XML
      - The full XML of the group
- $sources / $targets
  - Notable members:
    - Property: ParentWeb
      - The parent site collection SPWeb object of the group collection on the respective server

Items

Item transformers execute during a list item collection copy

Variables

- $dataObject
  - SPListItem
  - Notable members:
    - Property: XML
      - The full XML of the item
    - Property: XMLWithVersions
      - The full XML of the item with all versions included
    - Method: SetFullXML(XmlNode xNode)
      - A method that takes in an xmlnode and sets the item XML to it. Extremely useful to update the entire XML of an item.
    - Property: SpLIstItem[“Field Name”]
      - Retrieves the named field value from the list item. Can be set by assigning a value in a PowerShell assignment statement
      - Example: $dataObject[“Title”] = “New Title”
- $sources / $targets
  - Notable members:
    - Property: ParentList
      - The parent list of the item collection on the respective server
    - Property: ParentFolder
      - The parent folder of the item collection on the respective server

Lists

List transformers execute during a list collection copy

Variables

- $dataObject
    - Type: SPList
    - Notable members:
        - Property: ListXML
            - The full XML of the list
        - Property: ID
            - The list ID
        - Property: Name
            - The list name
        - Method UpdateSettings(string sXML)
            - A very useful method that allows you to update the entire list XML by calling it.
- $sources / $targets
    - Notable members:
        - Property: ParentWeb
            - The parent web of the list collection on the respective server

Permissions

Permission transformers execute during a permission collection copy

Variables

- $dataObject
    - Notable members:
        - Property: XML
            - The full xml of the permissions object in the collection

Site Columns

Site column transformers execute during a site level field collection copy

Variables

- $dataObject
    - Type: SPField
    - Notable members:
        - Property: ID
            - The ID of the field
        - Property: Name
            - The name of the field
        - Property: Type
            - The field type
        - Property: FieldXML
            - The full XML of the field.
- $sources / $targets

    - Notable members:
        - Property: ParentWeb
            - The parent web of the field collection on the respective server

Sites

Site transformers execute during a web copy

Variables

- $dataObject
    - Type: SPWeb
    - Notable members:
        - Property: XML
            - The full XML of the web object
        - Property: ServerRelativeURL
            - The server relative URL of the web
        - Property: DisplayURL
            - The full URL of the web
        - Property: SiteColumns
            - The field collection of site columns on the web
        - Property: RootSite
            - The root site collection object for the hierarchy.
        - Method: Update(string sXML, UpdateWebOptions options)
            - A very powerful method that allows updating of web properties and XML by specifying an XML block and specific update options.
- $sources / $targets
    - Notable members:
        - Property: ParentWeb
            - The parent web object of the web collection. If the web is the root site collection, this property returns itself

Users

User transformers execute during a referenced users and groups copy

Variables

- $dataObject
    - Type: SPUser
    - Notable members
        - Property: LoginName
            - The primary login name used for authentication for the user
        - Property: Email
            - The user email address
        - Property: SID
            - The unique identifier for the user
- $sources / $targets
    - Notable members:
        - Property: ParentWeb

- The parent web object of the user collection.

Web Parts

Web part transformers execute during a web part collection copy

Variables

- $dataObject
  - Type: SPWebPart
  - Notable members:
    - Property: XML
      - The full web part XML
    - Property: SPWebPart[“Property Name”]
      - Gets or sets the named property value
- $sources / $targets
  - Notable members:
    - Property: Parent
      - The parent web part page for the web part collection. The parent web can be retrieved by calling $dataObject.Parent.ParentWeb

ABOUT QUEST

Quest helps our customers reduce tedious administration tasks so they can focus on the innovation necessary for their businesses to grow. Quest® solutions are scalable, affordable and simple-to-use, and they deliver unmatched efficiency and productivity. Combined with Quest's invitation to the global community to be a part of its innovation, as well as our firm commitment to ensuring customer satisfaction, Quest will continue to accelerate the delivery of the most comprehensive solutions for Azure cloud management, SaaS, security, workforce mobility and data-driven insight.

If you have any questions regarding your potential use of this material, contact:

Quest Software Inc.
Attn: LEGAL Dept
4 Polaris Way
Aliso Viejo, CA 92656

Refer to our Web site (http://www.quest.com) for regional and international office information.