RedundancyMaster Help

Table of Contents

Help version 1.021

CONTENTS

Introduction

OPC Data Access (DA) Technology is one of the most successful developments made on the industrial automation software landscape. Virtually every application developed now uses some form of OPC DA technology. In this regard, OPC DA technology has proven to be reliable in virtually every possible situation that requires consistent data access to devices and systems. With the success of OPC technology, it is no wonder that so many applications and solutions have come to depend on the availability of data supplied by some form of OPC DA. Applications (such as data collection for compliance, alarm detection, product asset management, and human machine interface and SCADA) cannot afford to lose their connection to the underlying OPC server, target devices, or systems.

There are many factors that can impact the quality and reliability of the data. The OPC server may lose data when the following occurs:

1. The PC running the OPC server is shut down.
2. The user makes an error that causes the OPC server to exit.
3. The network connection to OPC server is lost or unreliable.
4. The network setting changes cause a link failure.
5. The OPC server fails for any reason.
6. The log-in changes on the OPC server's PC.

In most of these cases, the OPC DA server fails to provide data due to an actual failure underlying the OPC server or the connection to that server. These types of failures are called Object-based failures, and they occur when the actual link between the OPC client application and the target OPC server breaks down.

Loss of Data

Since communications have a number of factors that can dramatically affect reliability (especially in the industrial environment), the application can lose data in several ways. Examples of these physical factors are as follows:

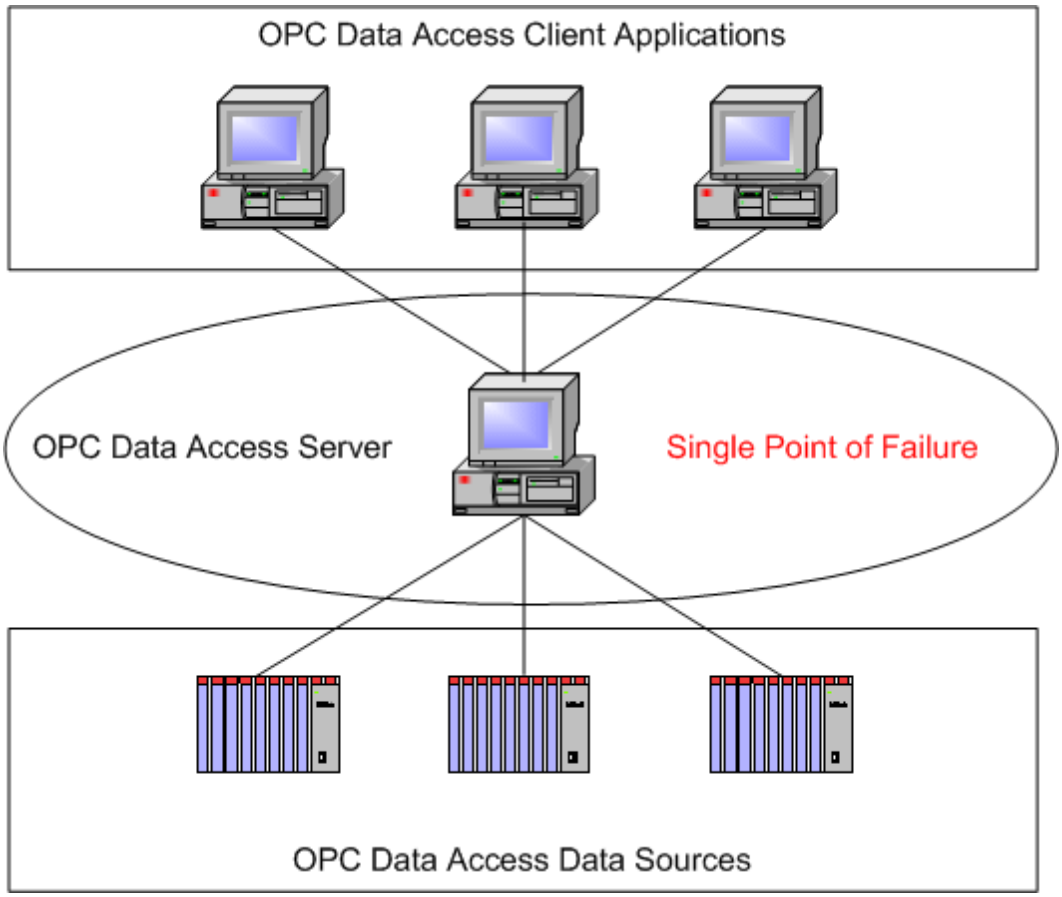
1. Physical connection failure. For example, the cable is pulled.
2. Hardware failure. For example, the router fails.
3. Electrical interference. For example, high current discharge.
4. Delays due to signal propagation. For example, radio links.
5. Environmental factors. For example, lightning.
6. Random accidents.

In these cases, the connection between the OPC server and the client may be perfectly intact but the physical link to the underlying device or system may be broken. These types of failures are called link-based failures, and they

occur when the connection to the target device or system has been lost. The OPC server in use is usually still completely operational, but simply cannot supply the data.

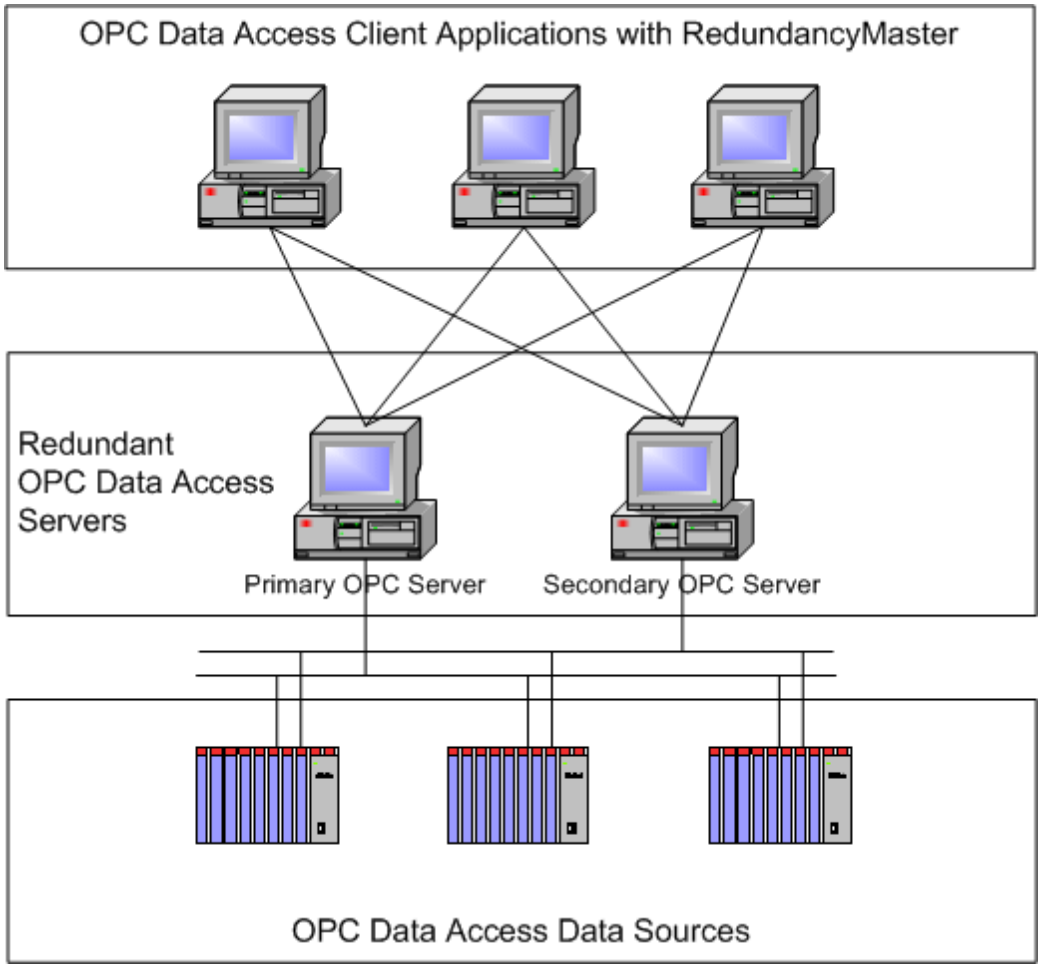
Increasing OPC Reliability

The following diagram demonstrates how a typical OPC application is configured and how it is susceptible to failure. As can be seen, the OPC DA client applications are all accessing a single OPC server; thus, the potential exists for both an Object-based failure and a Link-based failure. If for any reason the single OPC server fails to operate, then an Object-based failure occurs. Additionally, since this single PC is responsible for data collection from the underlying devices, a single point of failure exists for the device connection as well. To increase the reliability of the OPC system, users must remove these single points of failure.



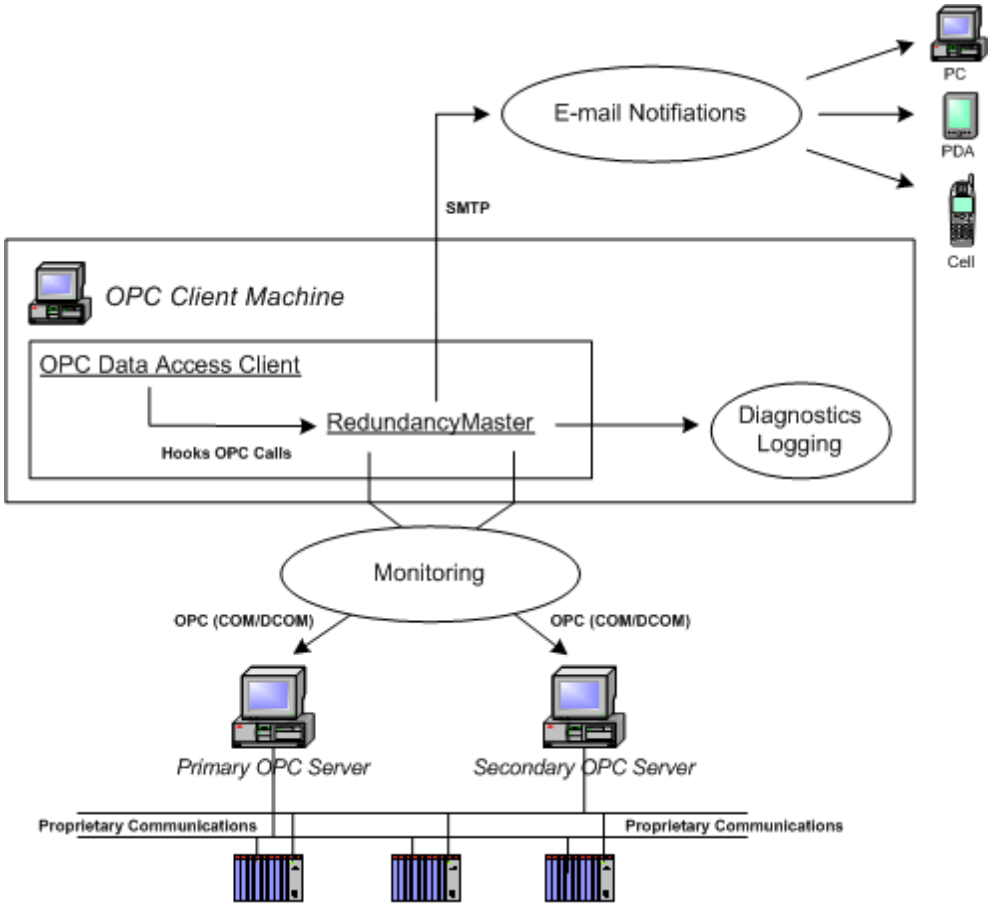
A Better Way

To eliminate the single point of failure, users can redesign the OPC applications to use more than one OPC server. The problem is that this requires the OPC client applications to have some form of redundancy built into the product. In either case, users may be required to redevelop the OPC client from the ground up to take advantage of the redundant technology. With an existing application of any considerable size, it may not be practical to completely redesign and reconfigure the application. Users need a seamless way to drop redundancy into the OPC application without making any changes to the OPC client.



As shown above, the original OPC application has been redesigned using two OPC servers instead of a single OPC server. To facilitate the redundant operation of the OPC servers, each OPC client has been paired with RedundancyMaster. The design of RedundancyMaster allows this to occur with no configuration changes to the OPC client application. Using the configurable options within RedundancyMaster, the use of either the Primary or Secondary OPC server can be controlled directly. Based on the modes selected, RedundancyMaster will keep both servers active or if configured to do so, start the secondary server only when the primary server fails.

In regard to either Object-based failures or Link-based failures, RedundancyMaster can be configured to monitor these conditions. Subsequent sections of this manual will detail how users can configure RedundancyMaster to monitor Object failures and Link failures (as well as what actions to take when they occur).



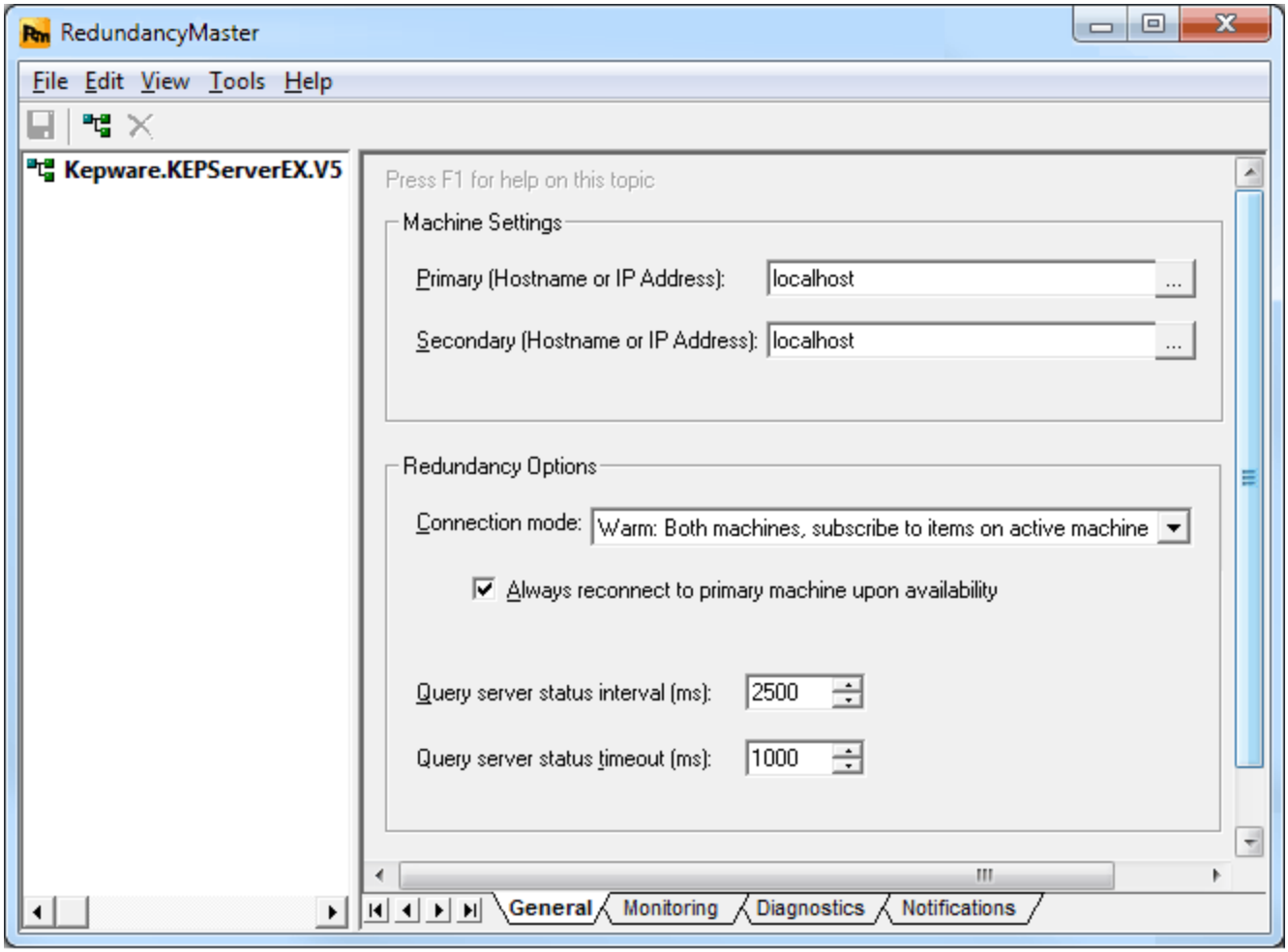
RedundancyMaster Architecture

RedundancyMaster's architecture seamlessly integrates redundancy. To prevent users from having to change the OPC client application, RedundancyMaster hooks all of the OPC calls that the application normally makes directly to an OPC server. Once the OPC calls have been hooked by RedundancyMaster, it can automatically route those calls to either the primary or secondary OPC server. The OPC client application never knows that this switching action occurs.

While it is important that the OPC client application has no idea that RedundancyMaster has switched between the primary or secondary OPC server, it is crucial to be aware that the switch has occurred since this may signal a possible loss of data if remedial action is not taken. To address this, RedundancyMaster logs all events that occur within its environment. RedundancyMaster can also generate email notifications for each logged event. To ensure that the email notifications go through, the Simple Mail Transfer Protocol (SMTP) servers allow for both a primary and secondary server to be defined.

User Interface

To learn more about RedundancyMaster's user interface and other specific elements, click on any area in the image below.



Requirements

Operating System

Because this application can run as an NT-based service, it is required to be installed on one of the following operating systems.

- Windows XP Service Pack 1 and 2
- Windows Server 2003 SP2*
- Windows Vista Business/Ultimate*
- Windows Server 2008/2008 R2*
- Windows 7*

*When installed on a 64 bit operating system, the application will run in a subsystem of Windows called WOW64 (Windows-on-Windows 64 bit). WOW64 is included on all 64 bit versions of Windows and is designed to make differences between the operating systems transparent to the user.

Note: RedundancyMaster is capable of providing redundancy to any OPC-compliant server running remotely on an OS platform. The OPC server must conform to the server side components required by the OPC Foundation.

See Also: **Installing as an NT Service**

Client System

This application must be installed on the same machine as the OPC clients that are being provided redundancy. This requirements is beneficial to the application by allowing it to provide redundancy to client projects without making any modifications to the client configuration.

OPC Servers

RedundancyMaster provides redundancy for OPC Data Access servers adhering to specifications 1.0, 2.05a, and 3.0. Note the following:

- The OPC Data Access servers can reside on any machine that is accessible from the system running RedundancyMaster.
- OPC servers are not limited to the operating systems listed above. Those listed above are required for RedundancyMaster.
- If an OPC server does not reside on the same machine as RedundancyMaster, a minimal OPC server configuration will be imported to the RedundancyMaster machine in order to enable redundancy.
- The OPC server must run as an EXE-based, out-of-process server or RedundancyMaster will not function properly.

RedundancyMaster can provide redundancy for multiple OPC servers and will support up to two machines per OPC server. The OPC server on each machine must use the same Prog ID and conform to the same OPC item namespace in order for redundancy to be achieved. For example, if the client requires an item named "Channel.Device.Tag," then this item must be configured on both machines.

See Also: **Adding Redundancy** and **Deploying the Project**.

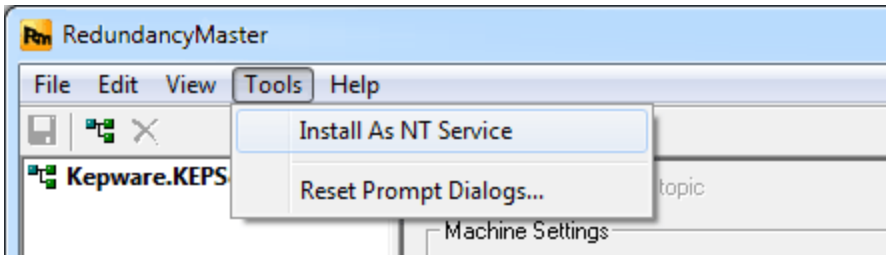
OPC Clients

RedundancyMaster provides redundancy for any client that can communicate with OPC Data Access servers adhering to specifications 1.0, 2.05a, and 3.0. The client must reside on the same machine as this application; that is, the machine on which RedundancyMaster is installed. It attaches itself to a client application as a DLL-based, in-process server. Therefore, the OPC client application must not prevent connections to a DLL-based, in-process server or RedundancyMaster will not function properly.

Installing as an NT Service

RedundancyMaster has the ability to run as a service. This ability is required for client applications that run as a service and continue to run when the machine experiences a user logout. If the client application does not run as a service, then users are not required to install this application as a service.

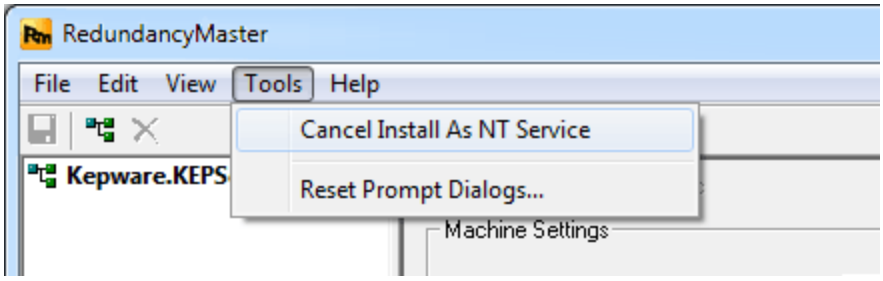
To install the application as a service, select **Tools** | **Install As NT Service**.



Note: Users must restart the application (as well as any clients that are currently utilizing redundancy) in order to complete the operation.

Canceling the Install as NT Service

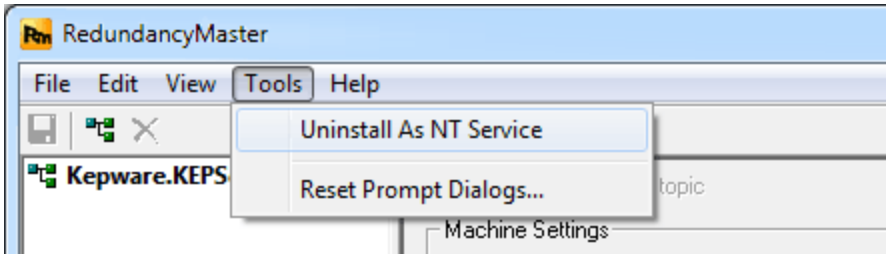
To cancel the request, select **Tools** | **Cancel Install As NT Service**.



Note: The application and clients do not need to be restarted after canceling.

Uninstalling as NT Service

To uninstall the application as an NT service, select **Tools** | **Uninstall As NT Service**.



Note: Users must restart the application (as well as any clients currently utilizing redundancy) in order to complete the operation.

Canceling the Uninstall as NT Service

If users decide not to uninstall the application as a service, they may cancel the request by selecting **Tools** | **Cancel Uninstall As NT Service**. The application and clients do not need to be restarted after canceling.

Setting Up Redundancy

For more information on a specific topic, select a link from the list below.

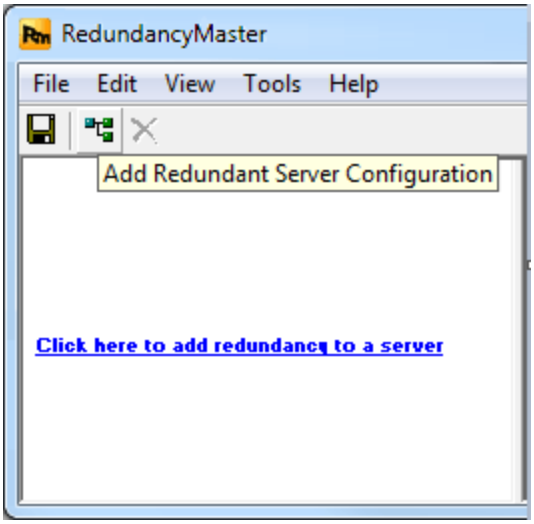
Adding Redundancy
Aliasing Redundancy
Unaliasing Redundancy
General Settings
Monitoring Settings
Diagnostics Settings
Notifications Settings
Applying Your Settings
Enabling and Disabling Redundancy
Removing Redundancy
Deploying the Project

Adding Redundancy

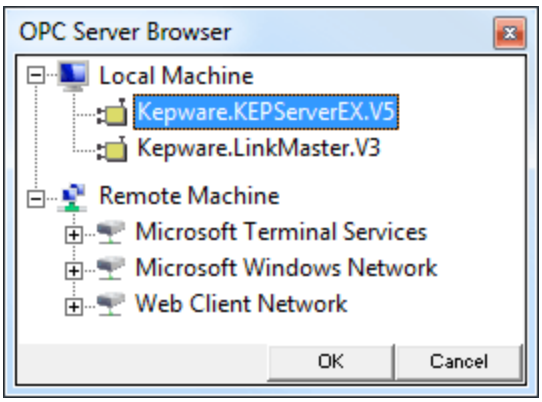
For information on adding redundancy to an OPC server, follow the instructions below.

Note: When creating multiple redundant pairs that will use the same OPC server, each redundant pair must be assigned a ProgID alias. For more information, refer to **Aliasing Redundancy**.

1. To start, verify that all system requirements have been met.
2. Next, click **Add Redundant Server Configuration**. Alternatively, select **Click here to add redundancy to a server**.

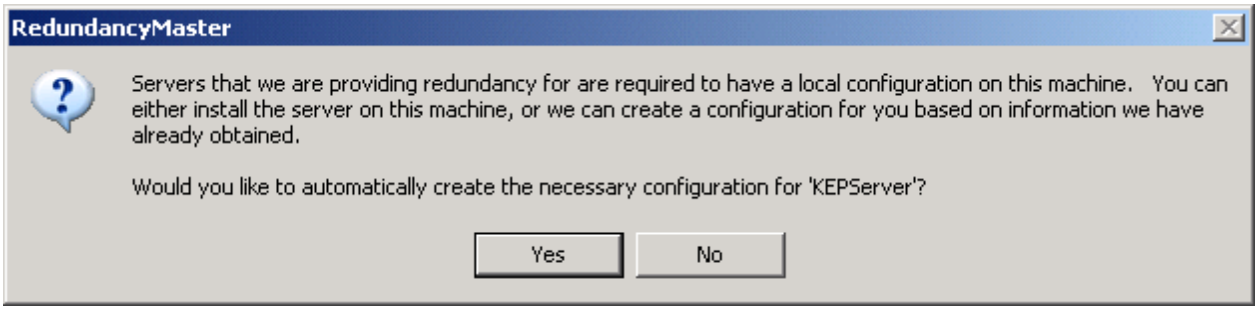


3. Next, locate and select the ProgID of the OPC server to which redundancy will be added. This OPC server may be on the local or remote machine.



4. Click **OK** to continue.
5. If the server is on a remote machine and is not installed on the local machine, a message window will be invoked. Click **Yes** to create the necessary configurations on the local machine.

   Note: If this redundant server configuration is removed (or if the RedundancyMaster application is uninstalled from the local machine), the minimal configuration for the remote server will be automatically cleaned up from the local machine. For more information, refer to **Removing Redundancy**.



6. To set up the minimal configuration requirements, select the **General Settings** tab. For more information, refer to **General Settings**.

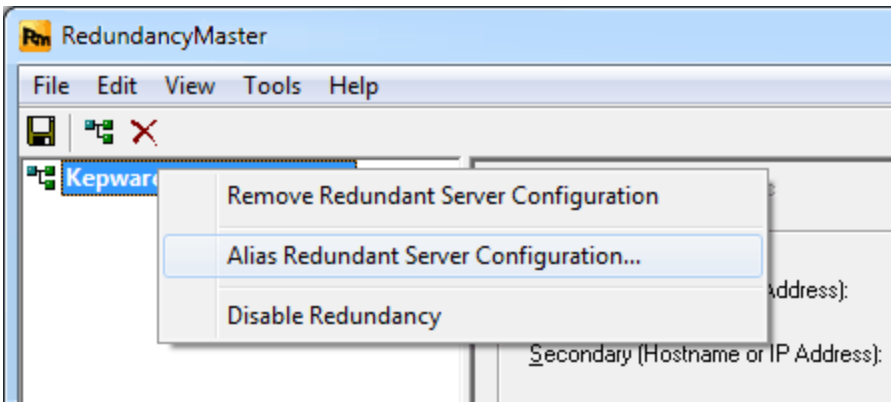
Important: Once changes have been made, users must remember to save the new settings. For more information, refer to **Applying Your Settings**.

Aliasing Redundancy

When creating multiple redundant OPC server pairs in RedundancyMaster that use the same ProgID, users must create an aliased ProgID for each of the redundant pairs. This allows OPC clients to connect to the specific redundant pair by referring to its aliased ProgID. For more information on aliasing an existing redundant pair, refer to the instructions below.

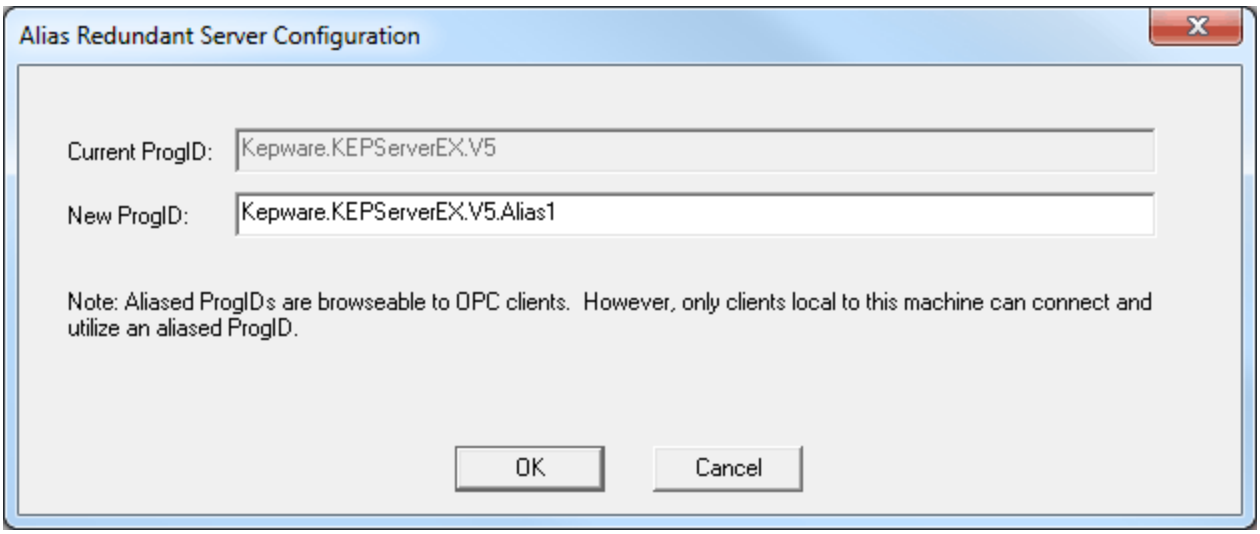
Note: If planning to use an existing OPC client project to connect to redundant pairs in RedundancyMaster that use aliased ProgIDs, users may be required to change the existing client project. In the client project, references to the original ProgID may need to be changed to the alias.

1. Create an aliased ProgID from an existing redundant pair's assigned ProgID. To do so, right-click on the existing ProgID and then select **Alias Redundant Server Configuration...**.

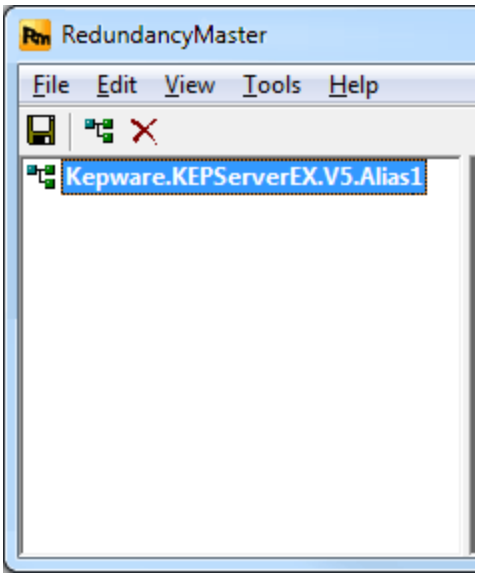


2. Next, click **OK** to accept the default ProgID alias. Alternatively, type a different name and then click **OK**.

   Note: ProgID alias names cannot include the characters <, > and &.



Note: The left pane should now display the ProgID with its alias.

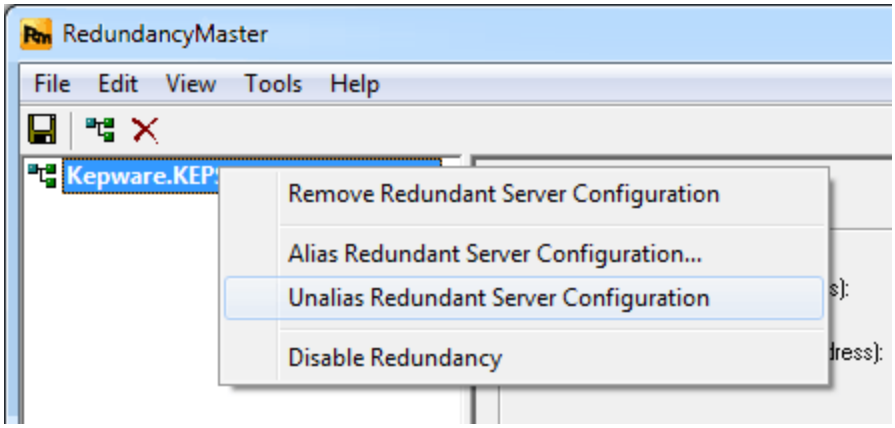


3. Once one redundant pair has been aliased, another redundant pair can be created. An alias can then be assigned to that pair (such as, *ServerProgID*.Alias2).

See Also: Unaliasing Redundancy

Unaliasing Redundancy

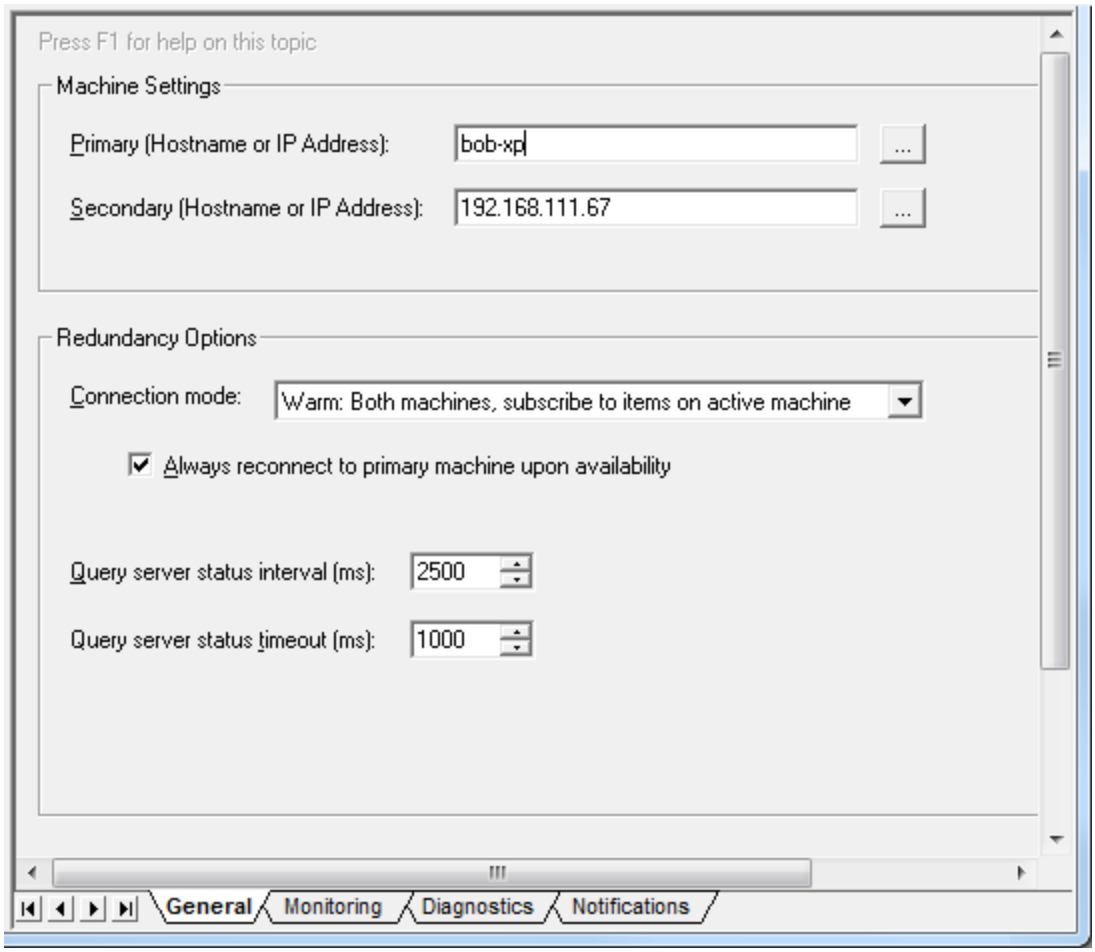
To remove an alias from a redundant pair, right-click on the alias in the left pane of the main RedundancyMaster screen and then select **Unalias Redundant Server Configuration**. Alternatively, click **Edit** | **Unalias Redundant Server Configuration**.



Note: Unalias is not allowed if the original ProgID of the alias is currently being used as a redundant server.

General Settings

The General Settings tab provides the minimal configuration requirements that must be defined in order to provide redundancy for the selected OPC server. These settings define the location of the underlying servers, describe how to connect to the servers, and describe how often to test communications with the servers.



Descriptions of the parameters are as follows:

- **Machine Settings:** This parameter defines the machines that the redundancy application should use in order to connect to the underlying primary and secondary servers.
- **Primary (Hostname or IP Address):** This parameter specifies the preferred connection that should be made to an OPC server. To set the primary machine's name, enter the host name or IP address into the

primary edit control. Alternatively, click the browse button to invoke the Machine Browser. If the primary machine is the same machine running this application, the machine name should be set to "localhost."

Note: Every time a new client connection is made to the underlying server, the application will attempt to connect to the server running on this machine. In the event that the connection to the primary fails or communications to the primary are lost, a connection to the secondary server will be attempted. Depending on the connection mode, users can configure the application to automatically establish communications with the primary machine when it becomes available. For more information, refer to **Connection Mode**.

- **Secondary (Hostname or IP Address):** This parameter specifies the fallback connection that should be made to an OPC server when communications to the primary machine are unavailable. To set the secondary machine name, manually type in the host name or IP address into the secondary edit control. Alternatively, click the browse button to invoke the Machine Browser. If the secondary machine is the same machine running this application, the machine name should be set to "localhost."

  Note: Depending on the connection mode, users can minimize fail-over time by configuring the application to establish a connection to the secondary machine even if the primary is available. For more information, refer to **Connection Mode**.
- **Connection Mode:** The connection mode defines when and how the redundancy application will connect to the underlying primary and secondary servers. The mode of operation affects the amount of time it takes to fail-over from one server to another. Some modes allow users to automatically promote communications to the primary when available. Options include Cold, Warm, and Hot. For more information, refer to "Connection Modes" below.
- **Always Connect to Primary Machine Upon Availability:** This parameter allows the redundancy application to automatically promote communications back to the primary machine when the OPC server becomes available. The default setting is checked.
- **Query Server Status Interval:** This interval specifies how often the redundancy application will ping the underlying servers to determine whether there has been a loss of communications. By querying at a faster rate, users can minimize fail-over time since failure detection occurs more frequently. The default setting is 2500 milliseconds.
- **Query Server Status Timeout:** This interval specifies how long the redundancy application will wait for a response from the underlying servers before considering there to be a loss of communications. The default setting is 1000 milliseconds.

Connection Modes

Cold: Active Machine Only

In this mode, the application will only connect to one underlying server at a time. On startup, a connection to the primary server will be made and all client related requests will be forwarded onto the primary. In the event that the connection to the primary fails (or communications to the primary are lost), a connection will be made to the secondary. If the redundancy application is unable to obtain a connection to the secondary, it will continue to alternate between the two servers until a successful connection is made. At that point, any subscriptions that the client has previously subscribed to will be automatically added to the newly 'active' server. Whenever communications to the active server are lost, the redundancy application will attempt to promote the inactive server in that alternating fashion.

Cold connection mode minimizes the amount of system resources that are allocated since only one connection is made to one server at any given time. This also reduces network traffic, since there is no need to poll the inactive machine as well as the active machine.

The amount of time it takes to fail-over to the inactive server is detrimental, however. When it is detected that communications with the active server have been lost, the application must establish the connection to the inactive server, subscribe to all items on behalf of the client, and then initiate the appropriate callback mechanisms. This time will vary depending on the underlying server (as well as network conditions). During this fail-over time, the client will not receive any data change or update notifications. Users should utilize one of the other connection modes if loss of data is crucial to the application.

Note: This mode does not support the "Always reconnect to Primary Machine" setting.

Warm: Both Machines Subscribe to Items on Active Machine

In this mode, the application will attempt to maintain a connection to both the primary and secondary servers at all times. On startup, the application will initialize data callbacks for the primary server only. In the event that the connection to the primary fails (or communications to the primary are lost), a data callback will be initialized for the secondary server. In any event, the application will only receive data from and/or write data to the active server.

Both servers will be pinged periodically to determine whether the connection is still valid. If the redundancy application loses communications to either server, it will periodically attempt to reconnect to the failed server. If the redundancy application is only communicating with one server, this will be considered the active server. Upon successfully connecting to the primary server, the primary will automatically be made the active server if "Always connect to primary machine upon availability" is selected.

Warm connection mode increases the amount of system resources that are allocated because two server connections are made on behalf of the client. There is also a minimal increase in network traffic because two servers are pinged periodically instead of one. The fail-over time is minimized, however, because the redundancy application only has to initialize data callbacks to the inactive server to begin receiving data. Users should utilize this connection mode to minimize network traffic and data loss.

Hot: Both Machines Subscribe to Items on Both Machines
In this mode, the application will attempt to maintain a connection to both the primary and secondary servers at all times. On startup, the application will initialize data callbacks for both primary and secondary servers so that both servers will send data change notifications. The data received from the primary will be forwarded onto the client. In the event that the connection to the primary fails (or communications to the primary are lost), data received for the secondary will be immediately forwarded onto the client. In either case, writes will only be forwarded to the active server.

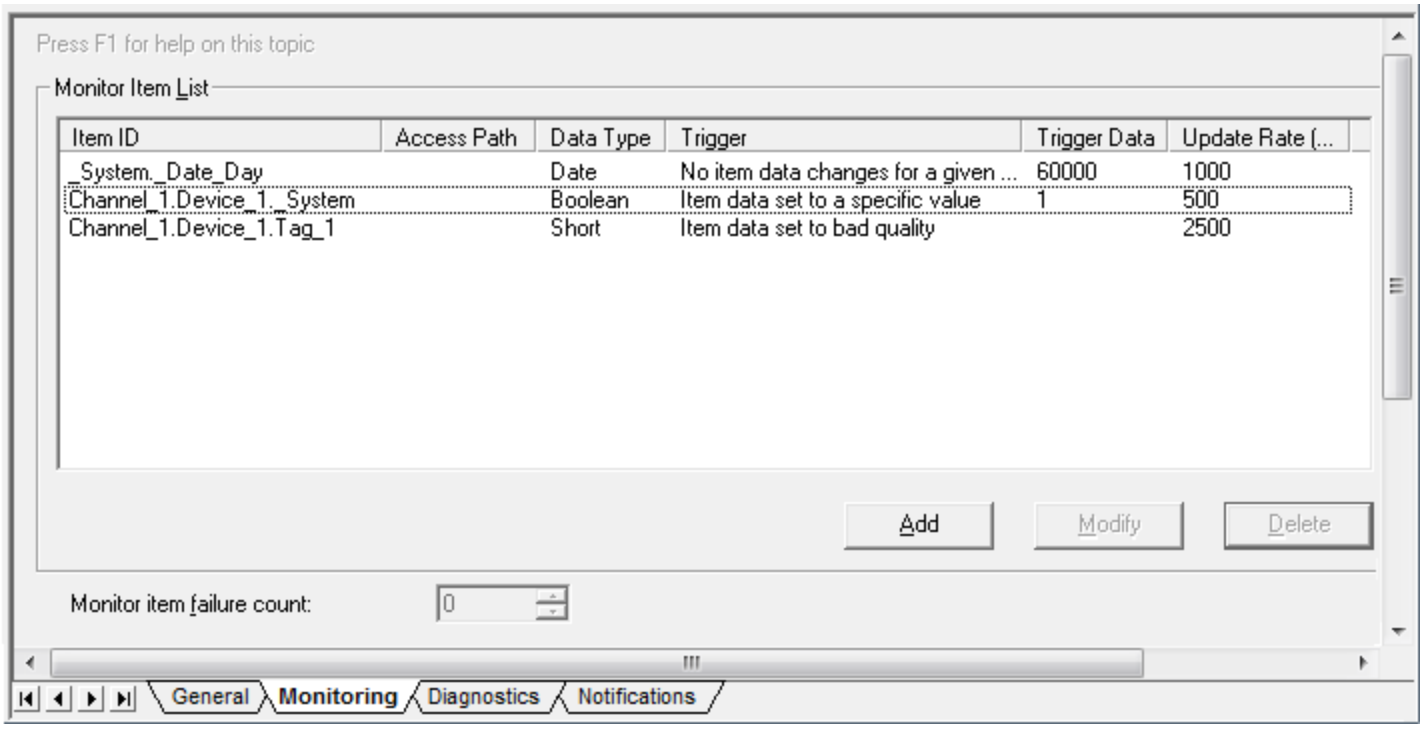
Both servers will be pinged periodically to determine whether the connections are still valid. If the redundancy application loses communications to either server, it will periodically attempt to reconnect to the failed server. Upon successfully connecting to the primary server, the primary will automatically be made the active server if "Always connect to primary machine upon availability" is selected.

Hot connection mode increases the amount of system resources that are allocated because two server connections are made on behalf of the client. There is also an increase in network traffic because data change notifications are being received from both underlying servers (and both servers are being periodically pinged to determine whether they are still available). This setting is useful in that fail-over time occurs immediately after detecting the loss of the active server. Users should utilize this connection mode if data loss is crucial to the application.

Important: Once changes have been made, users must remember to save the new settings. For more information, refer to **Applying Your Settings**.

Monitoring Settings

The Monitoring Settings tab specifies the conditions that will initiate a fail-over to the inactive server. These conditions can also be used to monitor server items for specific states in order to determine the health of the underlying servers/devices (beyond the automatic fail-over that occurs when communications are lost).

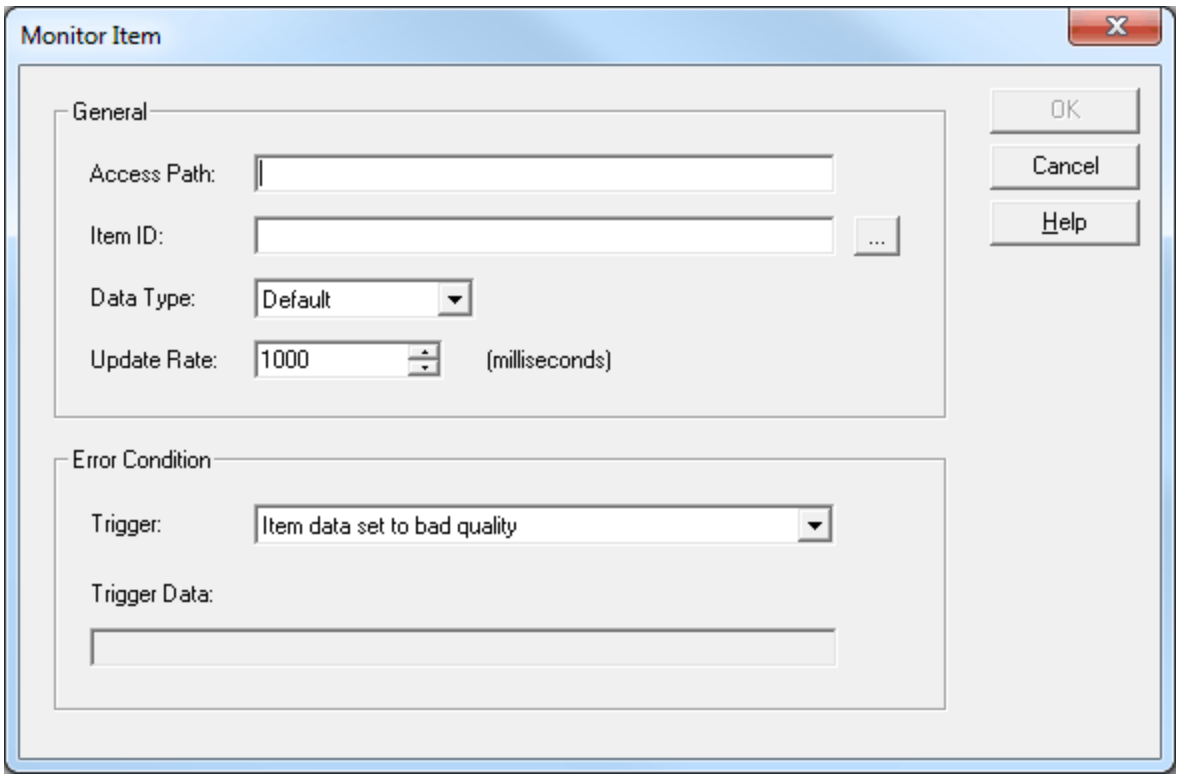


Descriptions are as follows:

- **Monitor Item List:** This list contains a collection of monitor items that can be assigned user-defined conditions. These items are monitored and tested to determine the health of the underlying servers and devices.
- **Monitor Item Failure Count:** This parameter specifies the maximum number of items in failure to count. If the number of items in error meets or exceeds this count, a fail-over to the inactive server will occur. This count can range from 1 to the number of monitor items configured.

Adding a Monitor Item

This dialog specifies the underlying OPC item definition, as well as the error condition for the monitor item. To add a monitor item, click **Add**.



Descriptions of the parameters are as follows:

- **Access Path:** This parameter specifies the access path for those servers that require or allow one as part of an OPC item definition. If the underlying OPC server supports item browsing, users can browse the server's name space by clicking the browse button instead.
- **Item ID:** This parameter specifies the fully-qualified path that defines an item in the underlying OPC server. For servers that use or require access paths, this may be a partially qualified string. Users can manually enter the string that defines the item in the Item ID edit control. Alternatively, users can also browse the server's name space by clicking the browse button.
- **Data Type:** This parameter specifies how the application will request the data format for the monitor item. Users can specify Default (which allows the server to return the data in the format of the item's native data type) or select a specific data type using the drop-down list. If the item definition is automatically filled in by using the item browse functionality, the data type selection will indicate the default data format.
- **Update Rate:** This parameter specifies how often the underlying server should attempt to provide the application with the latest data, quality, and time stamp for the monitor item. This rate is specified in milliseconds.
- **Trigger Condition:** This parameter specifies how the application will test the monitor item to determine if it is in error. There are currently three supported conditions. Descriptions of the conditions are as follows:

1. **Item data set to bad quality:** In this trigger condition, the item will be periodically tested to determine whether the quality of the data received by the underlying server is "bad". When a quality of bad is detected, this particular monitor item will be considered in error. Depending on the "monitor item failure count" setting, this error condition may force the application to fail-over to the inactive server. When a quality of "good" is detected, the monitor item error will be cleared.

2. **Item data set to a specific value:** In this trigger condition, the item will be periodically tested to determine whether the value of the data received by the underlying server is set to a specific value. Users must set the 'Trigger Data' to the specific value that they want monitored. When the specific value is detected, this particular monitor item will be considered in error. Depending on the "monitor item failure count" setting, this error condition may force the application to fail-over to the inactive server. When a value other than the specific value is detected, the monitor item error will be cleared.

   Note 1: Some servers allow users to set an item to a specific 'dead value' when in error. Users might consider using this particular test and set the trigger data to the appropriate dead value that will be reported by the server.

   Note 2: Array data for this particular trigger condition is not currently supported by this application.

3. **No item data changes for a given time (ms):** In this trigger condition, the item will be periodically tested to determine whether the server has sent a data change notification within a specific time period (specified in milliseconds). The trigger data must be at least two times the update rate associated with the monitor item. If the trigger data is less than two times the update rate, an informational prompt will notify the user that the value was auto-adjusted will be displayed. Users will need to set the 'Trigger Data' to above the maximum acceptable allowed time that can occur without a data change notification. When the allotted time expires, this particular monitor item will be considered in error. Depending on the "monitor item failure count" setting, this error condition may force the application to fail-over to the inactive server. Whenever an update is received, the monitor item error will be cleared and the expiration time will be reset.

Note 1: Most control systems implement error detection using watchdog values, which are values that change at a particular rate. If there are known values in the system that should be continuously changing, this trigger condition would be the most appropriate. Caution must be taken when selecting an appropriate timeout period. For example, even though a particular watchdog value may be changing every 100 milliseconds, the underlying server may not be able to poll this particular value at the same rate due to processing other tasks or polling other values. Users should determine the rate at which the underlying server can actually return a data change notification for this value to this application and then multiply it by 2.

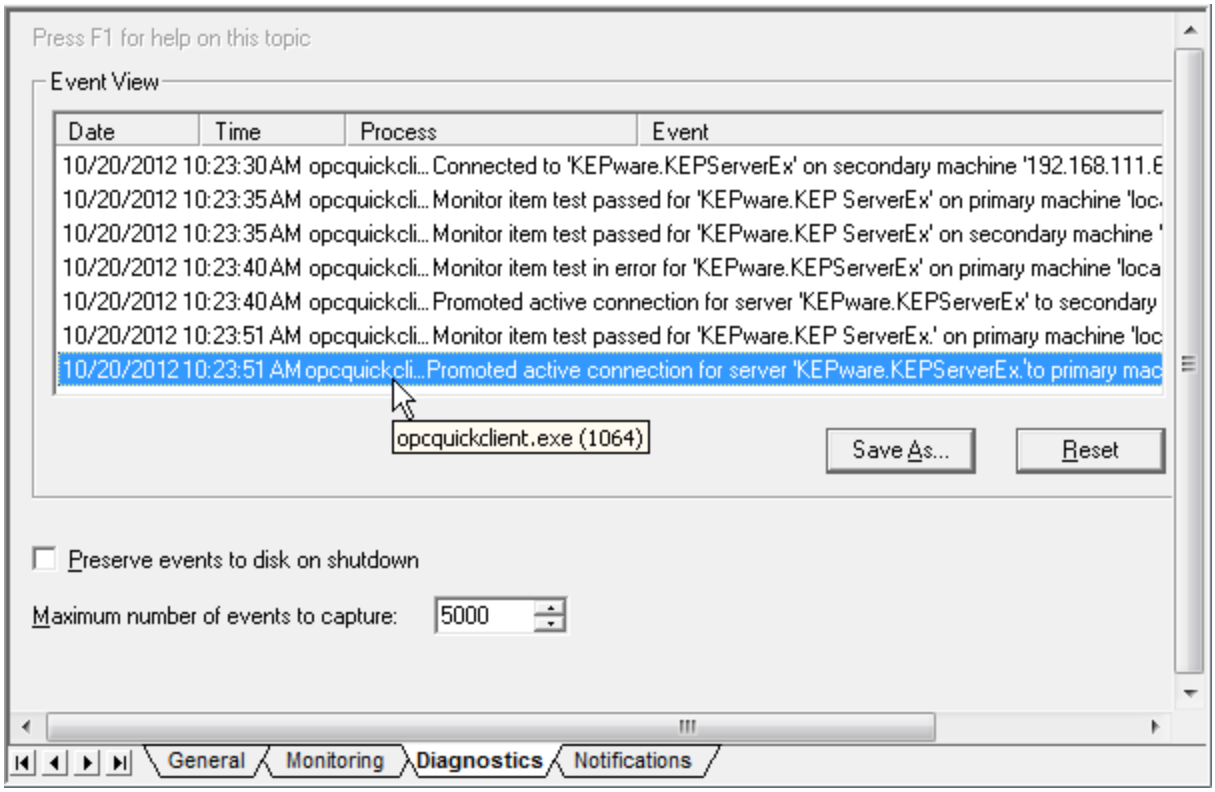
Note 2 : Items that cannot be added to the underlying servers will be considered in error. An appropriate error message that indicates the failure to add the monitor items will be logged.

Important: Once changes have been made, users must remember to save the new settings. For more information, refer to **Applying Your Settings**.

Diagnostics Settings

The Diagnostics Settings tab provides an event view that displays information or errors that have occurred during Runtime operations. Each event contains the date and time of occurrence, as well as the associated client process. For a comprehensive list of Runtime events, refer to **Runtime Diagnostics**.

Note: If the data in a particular column cannot be fully displayed, users can hover the cursor over the entry to invoke a tool tip that displays the full text.



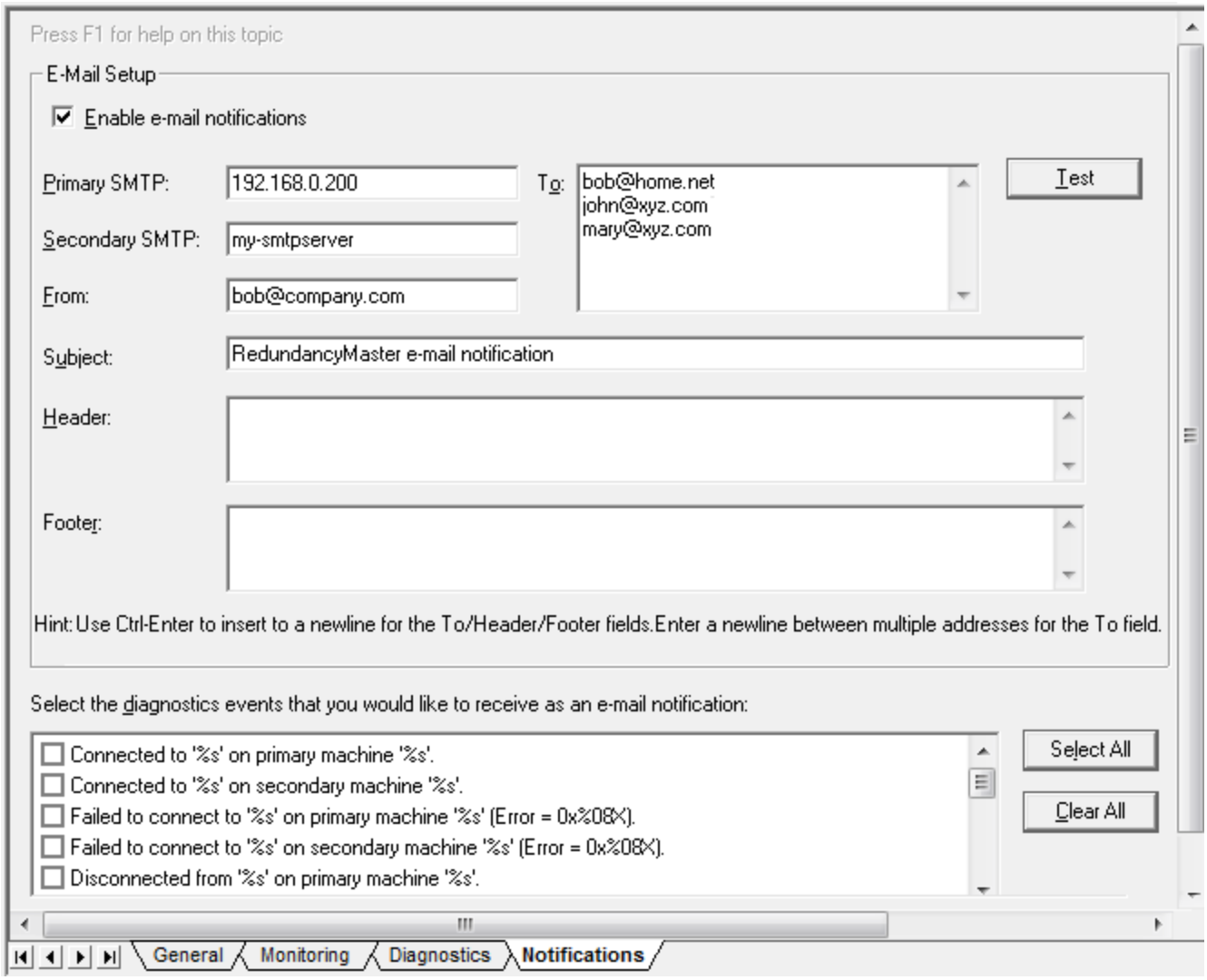
Descriptions of the parameters are as follows:

- **Save As:** When pressed, this button saves the current diagnostics to a text file.
- **Reset:** When pressed, this button clears the Event View and any diagnostics preserved to disk.
- **Preserve events to disk on shutdown:** When selected, events will be preserved to disk when the application is shutdown. The next time the application is started, the events will be displayed and any new events will be concatenated at the end.
- **Maximum number of events to capture:** This parameter specifies the maximum number of events to capture. Since diagnostics utilize memory and storage resources, users may want to limit the number of diagnostics that are preserved. Once the maximum number of events has been reached, the oldest events will be discarded as necessary. The valid range is 1000 to 30000. The default setting is 5000.

Important: Users must save the new settings after changes have been made. For more information, refer to **Applying Your Settings**.

Notifications Settings

The Notifications Settings tab specifies the recipients who will receive email notifications for one or more Diagnostics Events (which are visible beneath the local Diagnostics Settings View).



Descriptions of the parameters are as follows:

- **Enable Email Notifications:** This parameter enables the email notification system.
- **Primary SMTP:** This parameter specifies the host name or IP address of the preferred machine that is capable of sending email utilizing the Simple Mail Transport Protocol. To set the primary SMTP, enter the host name or IP address into the primary SMTP edit control.

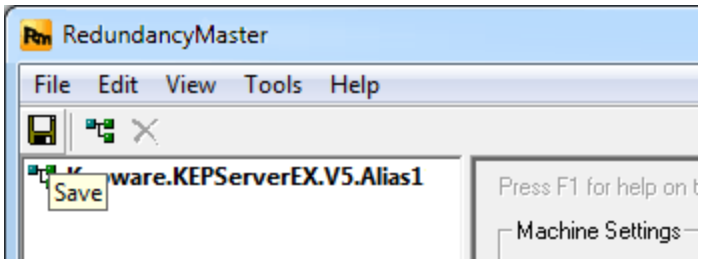
  Note: When an initial email notification is sent, the application will first attempt to send the notification through this machine. If the connection to the primary SMTP fails, the application will attempt to send the email notification through the secondary machine (if configured). If the email notification fails to be sent through both the primary and secondary machines, the email will be discarded and an error will be reported in the Diagnostics Event View. For more information, refer to **Diagnostics Settings**.
- **Secondary SMTP (Optional):** This parameter specifies host name or IP address of the secondary machine that is capable of sending email utilizing the Simple Mail Transport Protocol. To set the secondary SMTP, enter the host name or IP address into the primary SMTP edit control.
- **From:** This parameter specifies the email address of the sender that will appear as the originator of all email notifications sent from the application.
- **To:** This parameter contains the email address of one or more recipients to which all notifications should be sent. Each email address must appear on its own line. To start a new line, press **Ctrl+Enter.**
- **Subject (Optional):** This parameter contains the subject text that will appear in all email notifications sent from this application.
- **Header (Optional):** This parameter contains any text that will appear before the event text in the body of the email message.

- **Footer (Optional):** This parameter contains any text that will appear after the event text in the body of the email message.
- **List of Available Email Notifications:** The events selected in this list will be sent to recipients as they occur. The events that are available as email notifications are described in **Runtime Diagnostics**.
- **Test:** When clicked, an automated test message will be sent to all recipients.

Important: Once changes have been made, users must remember to save the new settings. For more information, refer to **Applying Your Settings**.

Applying Your Settings

RedundancyMaster can perform changes for many settings without having to stop and restart the client applications. The title bar will indicate that changes have been made to the settings by displaying "[configuration is modified]." In order to apply the changes, users must save the configuration. To do so, either click the **Save** toolbar button or click **File** | **Save**.

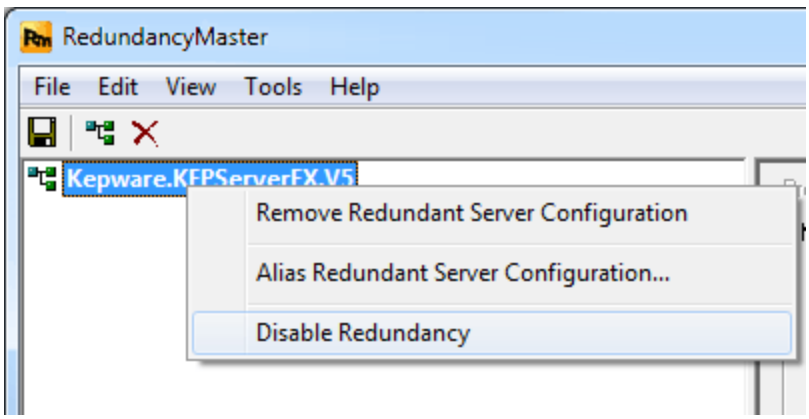


Note: If the configuration has been added and clients are running and connecting to the OPC server to which redundancy will be added, the client applications must be shutdown and restarted in order for redundancy to be initiated. Likewise, when removing redundancy from an OPC server that is currently being used, the client applications must be shutdown and restarted in order for redundancy to be fully removed.

Enabling and Disabling Redundancy

Disabling Redundancy

There may be times when redundancy should be unhooked from the client application. When doing so, it is not necessary to delete the redundant configuration from the application. To disable redundancy for a server, right-click on the server and then select **Disable Redundancy**.

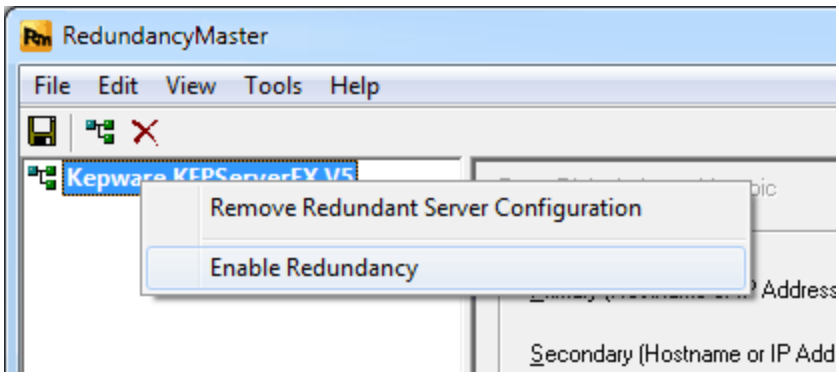


Note: The icon that was previously green will turn to gray to indicate its disabled state.

Important: Enabling and disabling redundancy on a server is not considered a configuration change because the required redundancy hooks must be immediately registered/unregistered (respectively) with the system registry. Users will not be prompted to save the configuration for this particular operation. As such, this is an operation that cannot be accomplished spontaneously. All client applications will have to be completely shutdown and restarted to ensure that redundancy is either enabled or disabled properly.

Enabling Redundancy

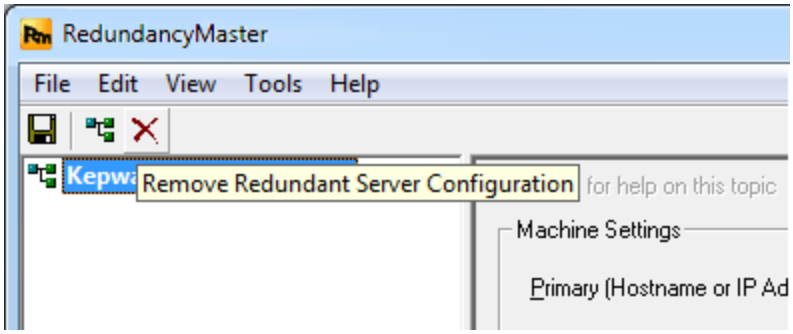
To re-enable redundancy, simply right-click the server as before and select **Enable Redundancy** from the context menu.



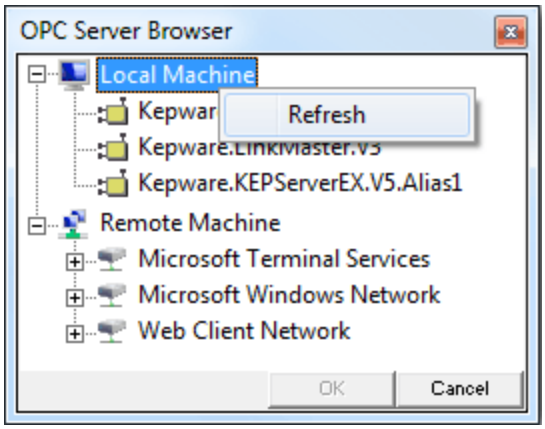
Note: The icon that was previously gray will turn to green to indicate its enabled state.

Removing Redundancy

To delete the redundant configuration for a server, select the server in the left-hand pane and then click the **Remove Redundant Server Configuration** toolbar button.



After a redundant server configuration has been removed, the redundant server will still be displayed in the OPC Server Browser window. To update the list of servers, click **Refresh**. To refresh the browser window, right-click on the machine name and then click **Refresh**. Afterwards, the redundant server that was removed should no longer be displayed.



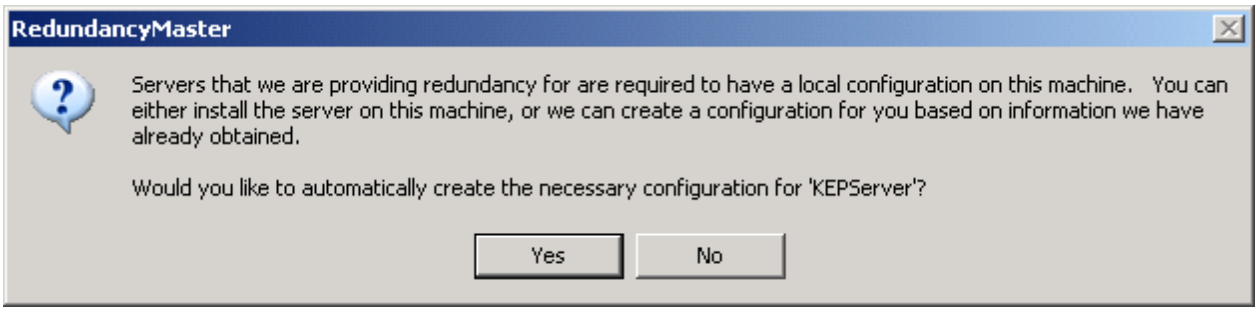
Important: Once changes have been made, users must remember to save the new settings. For more information, refer to **Applying Your Settings**.

Deploying the Project

There may be times where a redundancy project is configured and tested on one machine, but must be deployed on another machine (target). For information on deploying the project on another machine, refer to the instructions below.

1. Install the RedundancyMaster application on the Target Machine. The Target Machine must conform to the application requirements. If RedundancyMaster has been started on the Target Machine, shutdown the application before continuing.
2. Copy "redundancymaster.xml" file (located in the **\RedundancyMaster** installation directory of the test machine) to the **\RedundancyMaster** installation directory on the Target Machine.
3. Start RedundancyMaster on the Target Machine in order to activate the project that was copied.
4. All redundant servers will be disabled on the Target Machine. Users must enable these servers in order to provide redundancy to the underlying servers because the enable/disable state is machine-dependent and not contained in the configuration file. To enable the disabled redundant servers, right-click on the server name and then select **Enable Redundancy**.
5. If the server to which redundancy is being added is not installed on the local machine, a message window will be invoked. A minimal configuration for the remote server needs to be registered on the local machine in order for Redundancy Master to provide redundancy. Answer the prompt "Would you like to automatically create the necessary configuration for [*server name*]?" by clicking **Yes**.

   Note: If this redundant server configuration is removed (or if the RedundancyMaster application is uninstalled from the local machine), the minimal configuration for the remote server will be automatically cleaned up from the local machine.



6. In the confirmation message, click **Yes**.

Note: At this point, the project has been successfully deployed to the Target Machine. Any client applications that require redundancy will need to be restarted. Users should ensure that the appropriate security/DCOM security rights are assigned to the Target Machine.

See Also: **Enabling and Disabling Redundancy** and **Removing Redundancy**.

Runtime Diagnostics

RedundancyMaster logs events that provide description of information and errors that have occurred during Run-time. Events may be viewed in the diagnostics event view and may also be sent to one or more recipients as an email notification. Click on a link for a description of the event.

See Also: **Diagnostics Settings** and **Notifications Settings**.

Attempt to add monitor item '<item id>' for '<server name>' on primary machine '<machine name>' failed. This monitor item will be considered in error

Severity:
Error

Description:
A monitoring item that has been configured cannot be added to the underlying server on the primary machine. Items that cannot be added to the underlying servers will be considered in error. An appropriate error message will be logged that indicates the failure to add the monitor items.

Attempt to add monitor item '<item id>' for '<server name>' on secondary machine '<machine name>' failed. This monitor item will be considered in error

Severity:
Error

Description:
A monitoring item that has been configured cannot be added to the underlying server on the secondary machine. Items that cannot be added to the underlying servers will be considered in error. An appropriate error message will be logged that indicates the failure to add the monitor items.

Connected to '<server name>' on primary machine '<machine name>'

Severity:

Informational

Description:
A successful connection has been made to the underlying server on the primary machine.

Connected to '<server name>' on secondary machine '<machine name>'

Severity:
Informational

Description:
A successful connection has been made to the underlying server on the secondary machine.

Disconnected from '<server name>' on primary machine '<machine name>'

Severity:
Informational

Description:
The redundancy application dropped the connection to the underlying server on the primary machine.

Disconnected from '<server name>' on secondary machine '<machine name>'

Severity:
Informational

Description:
The redundancy application dropped the connection to the underlying server on the secondary machine.

Failed to connect to '<server name>' on primary machine '<machine name>'

Severity:
Error

Description:
The redundancy application cannot connect to the underlying server on the primary machine. Even though the redundancy application will continue to attempt this connection (depending on the connection mode settings), this error is not continuously posted: it is only posted on the initial attempt or on the transition from a connected to a failed state.

Failed to connect to '<server name>' on secondary machine '<machine name>'

Severity:
Error

Description:
The redundancy application cannot connect to the underlying server on the secondary machine. Even though the redundancy application will continue to attempt this connection (depending on the connection mode settings), this error is not continuously posted: it is only posted on the initial attempt or on the transition from a connected to a failed state.

Monitor item '<item id>' for '<server name>' on primary machine '<machine name>' contains array data which is not supported for the 'specific value' trigger condition

Severity:
Error

Description:
A monitoring item that has been configured with the 'specific value' trigger condition received array data from the underlying server on the primary machine. This application does not support array data for this particular trigger condition.

Monitor item '<item id>' for '<server name>' on secondary machine '<machine name>' contains array data which is not supported for the 'specific value' trigger condition

Severity:
Error

Description:
A monitoring item that has been configured with the 'specific value' trigger condition received array data from the underlying server on the secondary machine. This application does not support array data for this particular trigger condition.

Monitor item test in error for '<server name>' on primary machine '<machine name>'

Severity:
Error

Description:
The monitoring item test failed for the underlying server on the primary machine. This application will fail-over to the secondary machine. Even though the monitor item test may continue to fail for a given period of time, the redundancy application will not continuously post this error: it is only posted on the initial monitor test or on the transition to a failed state.

Monitor item test in error for '<server name>' on secondary machine '<machine name>'

Severity:
Error

Description:
The monitoring item test failed for the underlying server on the secondary machine. This application will fail-over to the primary machine. Even though the monitor item test may continue to fail for a given period of time, the redundancy application will not continuously post this error: it is only posted on the initial monitor test or on the transition to a failed state.

Monitor item test passed for '<server name>' on primary machine '<machine name>'

Severity:
Informational

Description:
The monitoring item test succeeded for the underlying server on the primary machine. This application will promote the primary machine, as required by the connection mode settings. Even though the monitor item test may continue to pass for a given period of time, the redundancy application will not continuously post this message: it is only posted on the initial monitor test or on the transition to a passed state.

Monitor item test passed for '<server name>' on secondary machine '<machine name>'

Severity:
Informational

Description:
The monitoring item test succeeded for the underlying server on the secondary machine. Even though the monitor item test may continue to pass for a given period of time, the redundancy application will not continuously post this message: it is only posted on the initial monitor test or on the transition to a passed state.

Promoted active connection for server '<server name>' to primary machine '<machine name>'

Severity:
Informational

Description:
The redundancy application successfully failed-over to the underlying server on the primary machine.

Promoted active connection for server '<server name>' to secondary machine '<machine name>'

Severity:
Informational

Description:
The redundancy application successfully failed-over to the underlying server on the secondary machine.

Received shutdown notification from server '<server name>' on primary machine '<machine name>'

Severity:
Informational

Description:
The redundancy application received an OPC shutdown notification from the underlying server on the primary machine. This notification is server specific and usually occurs if the server application is terminated by a user, the server application demo period times out or the machine the server is running on is shutdown. In either case, this application will then fail-over to the secondary machine. An attempt to reconnect to the primary will occur, as required by the connection mode settings.

Received shutdown notification from server '<server name>' on secondary machine '<machine name>'

Severity:
Informational

Description:
The redundancy application received an OPC shutdown notification from the underlying server on the secondary machine. This notification is server specific and usually occurs if the server application is terminated by a user, the server application demo period times out or the machine the server is running on is shutdown. In either case, this application will then fail-over to the primary machine. An attempt to reconnect to the secondary will occur, as required by the connection mode settings.

Unable to retrieve status for server '<server name>' on primary machine '<machine name>'. Server communications has been lost

Severity:
Error

Description:
The redundancy application failed to receive a ping response from the underlying server on the primary machine after establishing a connection successfully. This application will then fail-over to the secondary machine and attempt to reconnect to the primary, as required by the connection mode settings.

Unable to retrieve status for server '<server name>' on secondary machine '<machine name>'. Server communications has been lost

Severity:
Error

Description:
The redundancy application failed to receive a ping response from the underlying server on the secondary machine after establishing a connection successfully. This application will then fail-over to the primary machine and attempt to reconnect to the secondary, as required by the connection mode settings.

Index

A

C

D

E

F

G

I

M

N

P

R

S

T

U