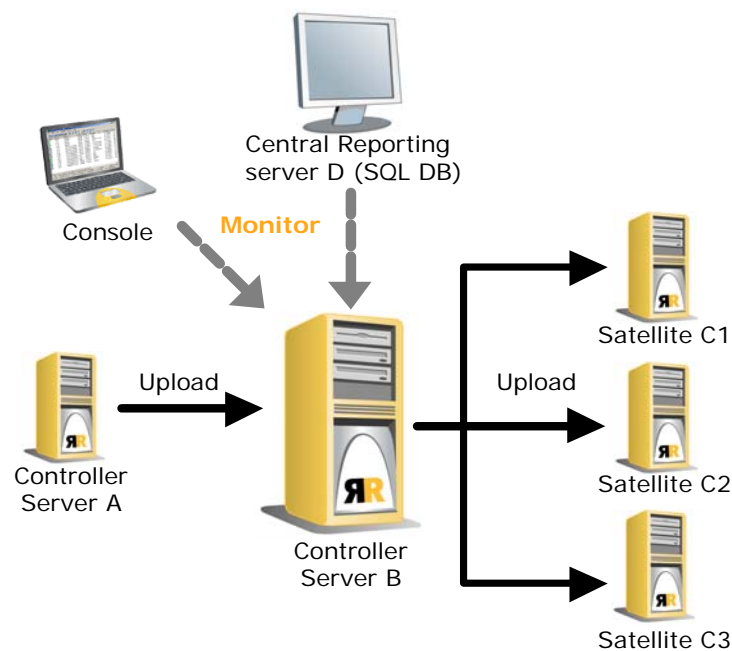


## RDS Centralized Reporting

Whether your network is small or large, the need for accurate and reliable reporting of what data has been transferred over the network has become a business critical operation.

In the example below we have a simple replication model where: Server A replicates to Server B that then updates Servers C1, C2, and C3



You then have a Central Reporting Server D running the SQL database, logging all the transfers.

Server D is separated from your replication topology but has data on all files and transfers throughout your network.

You can then query the SQL database for multiple tasks:

- Tracking - The progress of individual files can be tracked through your network. i.e. "Has all my web content been uploaded to all three web servers?"
- Performance Analysis - Query specific parts of the network and compare data throughput.
- Capacity Planning – Highlight bottlenecks or areas of under capacity for optimizing networks.

RepliWeb RDS has four levels of reporting on the job level as defined within the General Properties of a job. A special SQL database style reporting mechanism has been developed to collect and analyze the data in a central format

## 1. Job Configuration

On the completion of a job we define an 'On Exit' command that sends an email to the SQL database with all the data required.

1. Set the properties **Transfer Report Style** under the **General** Tab to either **Summary** or **Detailed**. Any other report style will result in an **empty** file list being created.

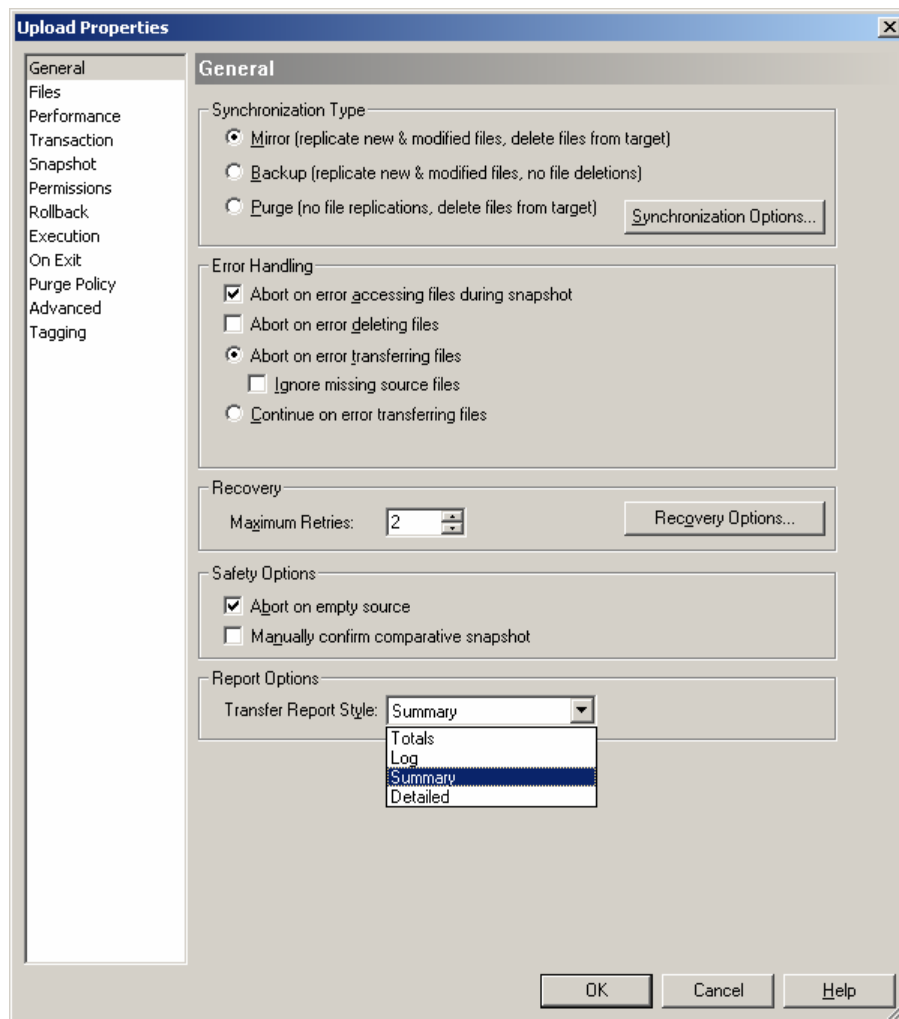
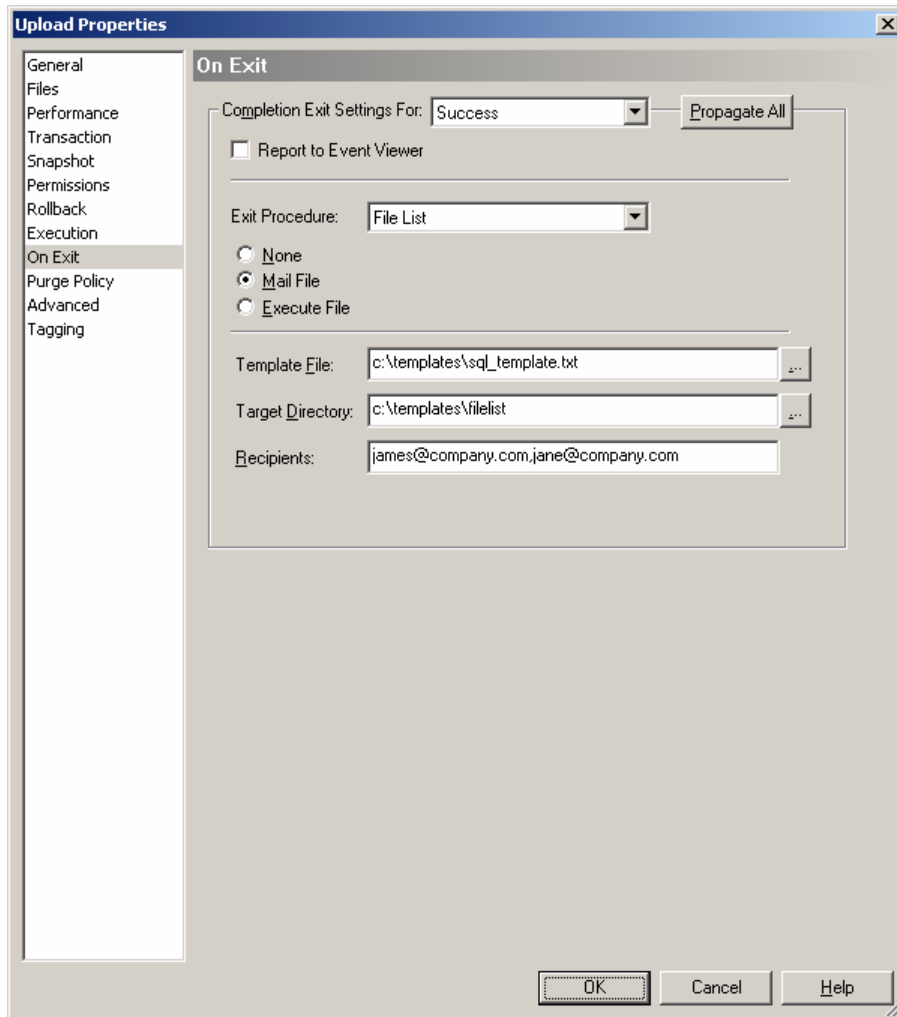


Figure 1 – General Tab – Transfer Report Format

## 2. Specify the File List properties in the **On Exit** tab



**Figure 2 – On Exit – File List**

More than one recipient may be used, separated by a comma.

The transfer reports are parsed according to the template file. The output file will be created in the Target Directory specified, and then emailed to the recipients.

## 3. Specify the File List properties

- **None** – Creates the file list on the Controller.
  - a. **Template File** – Mandatory.

The full path to a custom template must be defined. Templates are explained in detail in the following section.
  - b. **Target Directory** – Mandatory.

Specifies the directory on the Controller where the file list will be created. Do not specify a file name, as the file list will be created as

`file_list_JOBID.txt`, where *JOBID* is the unique id of the replication job.

- **Mail File List** – Mails the file list to the specified recipients
  - a. **Template File** – Mandatory.  
The full path to a custom template must be defined. Templates are explained in detail in the following section.
  - b. **Target Directory** – Optional.  
Use this feature to keep a copy of the File List on the Controller. Specifies the directory on the Controller where the file list will be created. Do not specify a file name, as the file list will be created as `file_list_JOBID.txt`. *JOBID* is the unique id of the replication job.
  - c. **Recipients** – Mandatory.  
Specify the e-mail address(es) of those receiving the file list. Multiple email addresses should be separated by a comma.  
i.e. `user1@company.com,user2@company.com`
- **Execute File List** – Creates a Windows batch file or UNIX script based on the specified template file. Since each line in the file list represents a unique file that was transferred, when the batch file / script is executed on the Controller, each line is executed in sequence. Some examples of using the **Execute File List** option are provided below.
  - a. **Template File** – Mandatory.  
The full path to a custom template must be defined. Templates are explained in detail in the following section.
  - b. **Target Directory** – Optional.  
Use this feature to keep a copy of the File List batch file or script on the Controller. Specifies the directory on the Controller where the file list will be created. Do not specify a file name, as the file list will be created as `file_list_JOBID.bat` on windows, and `file_list_JOBID.sh` on UNIX. *JOBID* is the unique id of the replication job.

For the Execute option, it is ideal to leave this field empty and use the default Job directory. That way the intermediate file will be purged along with the job itself.

More than one recipient may be used, separated by a comma.

The transfer reports are parsed according to the template file. The output file will be created in the Target Directory specified, and then emailed to the recipients.

## 2. Template File

A critical aspect of enterprise level data deployment is the ability to easily determine exactly which files were transferred to which systems, and the details associated with the transfer itself. For example, this capability is important for data integrity validation and to governance issues such as Sarbanes-Oxley. The RepliWeb RDS **File List** feature enables RDS deployment or replication to automatically create a file containing the above-mentioned information.

### File List Concept - A Multi-Steps Process

During the transfer stage a detailed **Transfer Report** is created, containing information about the files that were transferred in the replication process.

Using a text **Template** file, a **File List** file is created on the Controller, combining the information from the **Transfer Report**, with the format defined in a **Template** file. The Template file can be easily created in accordance with a user-defined format.

In the **File List** file, each line represents information pertaining to a single file transferred. Once the File List file has been created, it can then be stored locally on the Controller, e-mailed to multiple recipients, or even executed as a Windows batch file or UNIX script on the RDS Controller.

From an RDS perspective, the order of events is as follows:

1. The user creates a Template file, using environment variables provided by RDS.
2. The job is run with **On Exit – File List** option set. The job creates the log of transferred files (Detailed or Summary Transfer Report).
3. Upon job completion the job uses the Template file to create the File List file where each line represents a file that was transferred.
4. The File List file is emailed or executed on the Controller.

---

**NOTE:** The whole process, starting at creating the Template file, until the File List file is stored or executed, is performed exclusively on the Controller.

---

---

**NOTE:** Set the properties **Transfer Report Style** under the **General** Tab to either **Summary** or **Detailed**. Any other report style will result in an **empty** file list being created.

---

The template file dictates the format of the File List that will be created or executed (as explained above).

## Creating a Template File

The template file format is as follows:

**HEADER**  
*multiple lines written to the file once*  
**BODY**  
*one line written to the file for each file transferred*  
**FOOTER**  
*multiple lines written to the file once*

1. Section names are case sensitive
2. HEADER section may contain multiple lines.
3. HEADER section is optional. If omitted, there's no need to write the BODY header. The first line in the file will be regarded as the template line.
4. If BODY is specified, FOOTER must exist (but can be empty).
5. BODY may contain only one (1) line. Since each line in the file list represents a unique file transferred, the line in the template file is a representation of what each line in the file list will look like.
6. The template file line can contain any text, plus the following variables. When the file list is created after a deployment / replication, the variables are replaced on a line-by-line basis with the information about the given file.
7. Template line may use pre-defined variables for information about the transferred file:

RW\_JOB\_UNIQUE\_ID            Job ID  
RW\_JOB\_NAME                Job Name  
RW\_JOB\_COMPLETION\_STATUS   The job's completion status: SUCCESS|ABORT|ERROR

%j job name  
%c source hostname  
%n destination node/hostname  
%z file size (bytes)  
%s source file specification  
%d target file specification  
%t file transfer completion time  
%q sql time format file transfer completion time

### 3. Centralized Reporting

The template file, `c:\templates\sqltemplate.txt` can then be an SQL statement with `%cnzsdtq` in the relevant places, and once the transfer report has been parsed it is then emailed to the address defined in the registry key.

#### Sample of template file

```
INSERT INTO File_list
(Source_Hostname, Destination_Hostname, Filesize, Source_Filespec,
Target_Filespec, Tranfer_Completion_Time,
SQL_Time_Format_Transfer_Completion_Time)
VALUES ('%c', '%n', '%z', '%s', '%d', '%t' , '%q')
```

Where: File\_List **Table Name**  
Source\_Hostname  
Destination\_Hostname,  
Filesize,  
Source\_Filespec,  
Target\_Filespec,  
Tranfer\_Completion\_Time,  
SQL\_Time\_Format\_Transfer\_Completion\_Time **Fields Name**

Your SQL server can then pop this email box every x minutes and import the data.

This database then has all the data for all files that are transferred; individual files can be traced in distribution configurations or data throughput on any given server can be measured. All this reporting is on a post job completion basis, collecting data on what has been transferred when and between which servers.

Below is a sample of SQL script that checks new mail and inserts the data into the relevant table.

## SQL script Sample

```
EXEC xp_startmail 'mapi_profile_name'

declare @status int,
declare @hMessageID varchar(255),
declare @MailMessage varchar(255)
declare @filename varchar(255)
declare @mapifailure int
declare @msgsubject varchar(255)
declare @subject varchar(255)
declare @msg_id varchar(64)
declare @count int
declare @originator varchar(255)
declare @attachments varchar(255)
declare @rename varchar(255)
declare @query varchar(375)
declare @str varchar(255)

set @count=0

EXEC @status = xp_findnextmsg @msg_id = @msg_id output,
@unread_only='true'

while (@status=0)begin
    set @count = @count +1
    if @msg_id is null break
    if @msg_id = '' break

    exec @status = xp_readmail
        @msg_id = @msg_id
        ,@suppress_attach = 'false'
        ,@peek='false'
        ,@subject = @msgsubject output
        ,@originator=@originator output
        ,@attachments = @attachments output
    set @rename = 'copy /Y ' + @attachments + ' C:\Working
Directory\rds.sql'

    exec xp_cmdshell @rename
    select @str = 'isql -S server_name -E -i input_file
exec xp_cmdshell @str

    if (@status <> 0 ) begin
        print 'status from xp_readmail failed'
        break
    end

EXEC @status = xp_findnextmsg @msg_id = @msg_id
output,@unread_only='true'

end
EXEC xp_stopmail
```

## 4. Examples of Execute Template Files

An execute template file takes on the same form, and uses the same variables as a normal template file, but must contain commands that can be executed from a command-prompt / shell.

### Executing scripts and File List

To even enhance the user's power at the exit stage, this file can also include header and footer that are in free format, allowing calling other batch file or contain executable commands. This allows both File List and Post Commands as the exit procedure.

The File List Template can then look like this:

```
HEADER
echo Start of Job %RW_JOB_UNIQUE_ID%>> c:\lists\file_list.txt
echo Completion Status: %RW_JOB_COMPLETION_STATUS% >> c:\lists\file_list.txt
d:\repliwebbatchfiles\checkstatus.bat
BODY
echo %c,%n,%z,%s,%d,%t,%q >> c:\lists\file_list.txt
FOOTER
echo End of Job %RW_JOB_UNIQUE_ID%>> c:\lists\file_list.txt
```

As part of the replication process, at the exit stage, a batch file is created at the job directory, based on the template file. Then the batch file is executed, writing the Job ID and Completion Status using Environment variables, handling the job's completion status (d:\repliwebbatchfiles\checkstatus.bat) and writing the information about the transferred files to the log file (d:\lists\file\_list.txt). At the end of the list a line will be added marking the end of the list using the Environment Variable holding the Job ID.

The output file (c:\lists\file\_list.txt) will then look like this:

```
Start of Job 277
Completion Status: SUCCESS
srv,target_srv,1263,d:\source\record.info,d:\target\log\record.info,Tue
Feb 17 10 51 35 2004,2004-02-17 10:51:35
srv,target_srv,4320,d:\source\2\1\record.pack,d:\target\log\record.pack
,Tue Feb 17 10 51 35 2004,2004-02-17 10:51:35
srv,target_srv,126344,d:\source\2\2\record.info,d:\target\log\2\record.
info,Tue Feb 17 10 51 35 2004,2004-02-17 10:51:35
srv,target_srv,4320,D:\dmgr_source\2\2\record.pack,d:\target\log\record
.pack,Tue Feb 17 10 51 35 2004,2004-02-17 10:53:40
End of Job 277
```

## Delete source files after transfer



```
del "%s"
```

Would create the following batch file on the Center, which would then be executed when the job is complete:

```
del "C:\source\expenses.xls"  
del "C:\source\contract.doc"
```

## Command

In this example the attributes of each of the transferred files and append them to a file:



```
attrib "%s" >> C:\attributes.txt
```

Would create the following batch file on the Center, which would then be executed when the job is complete:

```
attrib "C:\source\expenses.xls" >> C:\attributes.txt  
attrib "C:\source\contract.doc" >> C:\attributes.txt
```

## Formatted HTML List

In this example, the file list takes on the following form:



```
<li><a href=%s>%s</a> Transferred at %t
```

Resulting in the following file that could be viewed in a web browser:

```
<li><a  
href="C:\source\expenses.xls">"C:\source\expenses.xls"</a>  
Transferred at Mon Jun 30 14 45 05 2003  
<li><a  
href="C:\source\expenses.xls">"C:\source\expenses.xls"</a>  
Transferred at Mon Jun 30 14 47 12 2003
```

###

**For any additional information, please contact us at [support.repliweb.com](http://support.repliweb.com)**