

# RDS Building Centralized Monitoring and Control

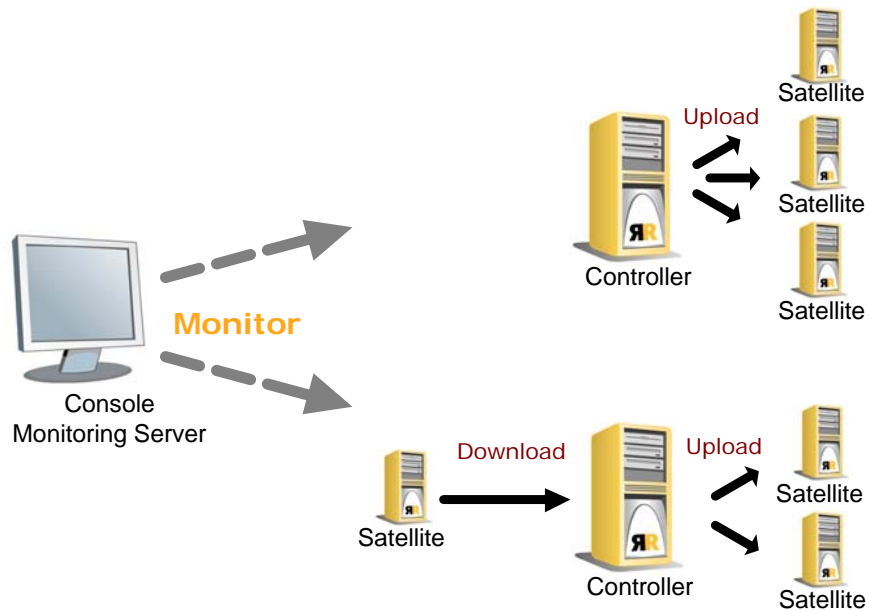
## 1. Overview

This document explains the concept and differing options for the monitoring and control of RDS replication over your network.

The very basic nature of RDS is a 3-tier architecture where a detached Console software component defines and monitors a replication process that takes place between a two other machines (A Controller and a Satellite). That architecture is preserved when doing central monitoring in the sense that the replication information is retrieved (by Console, Command Line, or API) from the Controller. To present a centralized picture of multiple replications taking place on multiple Controllers, multiple Controllers are accessed.

This document discusses the different approaches when having to set a single monitoring node that should monitor and control replication processes taking place between multiple controllers and multiple satellites.

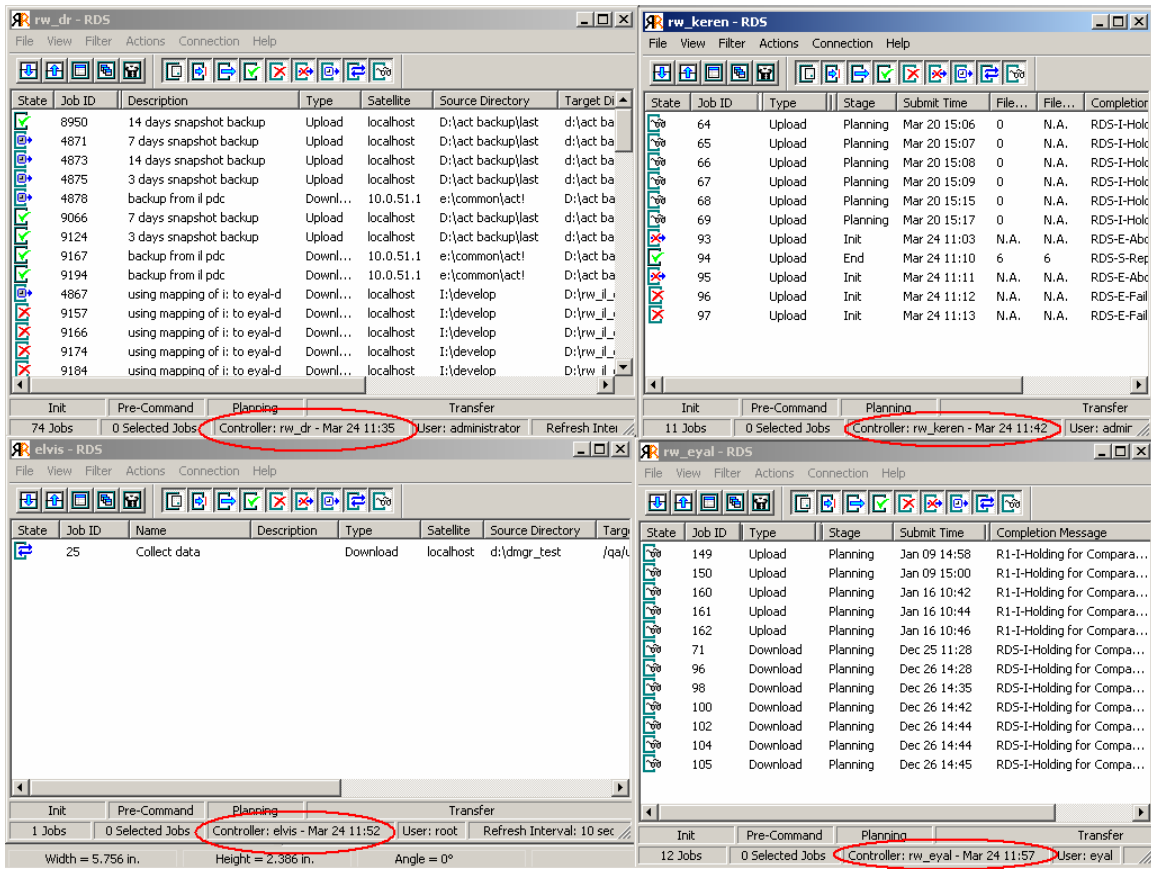
The following diagram describes the desired topology where replications take place between a variety of servers, and the monitoring node is not a part of the replication topology. The key point, as mentioned before is that the monitoring node needs to communicate (in a variety of ways that will be discussed) with the Controllers in the replication topology.



## 1. Using Multiple Consoles

This approach is recommended for small topographies or classic star configuration. Small topographies are in the sense of the numbers of Controllers involved in the replication, usually not more than 3 or 4, although the overall network can be a lot bigger. In these topologies, the simplest model is to have a multiple Consoles open, each connected to a different Controller.

The following example shows how a monitoring screen would look like and how the orientation is provided by the information in the GUI itself. Again, this is the simplest and most effective way when up to 3 or 4 controllers are involved. This is completely not practical when more Controllers are involved.



## 2. Polling Multiple Centers using RDS Command Line Interface

The key for understanding this approach is that the complete Console functionality of the GUI is available from the Command Line Interface (CLI).

As explained in the overview, the information about the replications taking place all over the network is contained in the Controllers of the network. In order to get a centralized picture, the information from all these Controllers should be gathered and consolidated into a central picture. This is done by building a simple batch file that connects in turn to each Controller and writes the job information from that Controller into an output file on the monitoring node.

An intrinsic benefit of the CLI over the GUI is the ability then to consolidate this output into one format. As apposed to the data being displayed in a graphical format of the GUI the information about jobs that is coming back from each Controller is written either in text format for simple presentation or in xml format so that another application can process it and build the presentation desired by the user.

An example for user implementation of this monitoring approach can be a pearl script running on a continuous basis, executing the CLI commands to pull the Controllers

information. That output of the CLI is processed to build a unified presentation for the end user. That unified presentation can be an output format suitable for a web browser or some other processing that will push the monitored information into a central monitoring framework that exists in the organization

The fact that the CLI can produce output in a tagged xml form, provides an immediate and easy way for parsing the CLI output without having to develop programs that use the API.

Appendix I of this document contains a simple example of building such a centralized monitoring script.

### **3. Building specific monitoring with the Application Programming Interface (API)**

Where using CLI gives the user an easy and simple way of generating the raw monitoring information, it can be complicated to build specific monitoring interfaces this way. In the term “specific” we refer to an enterprise need of giving a specific user specific, tailored, monitoring functionality over certain replication processes. In real networks, there can be multiple replication processes running in different or common parts of the network. They can be serving different needs like content deployment, or collection of data for disaster recovery. Even though they can be running on the same machines they are not connected, logically, and need to be monitored and controlled by different people. The specific monitoring can include the need to see different and customized information. In these cases, the approach should be to use the RDS Application Programming Interface (API) to poll the right information from the right controller and build the proper information.

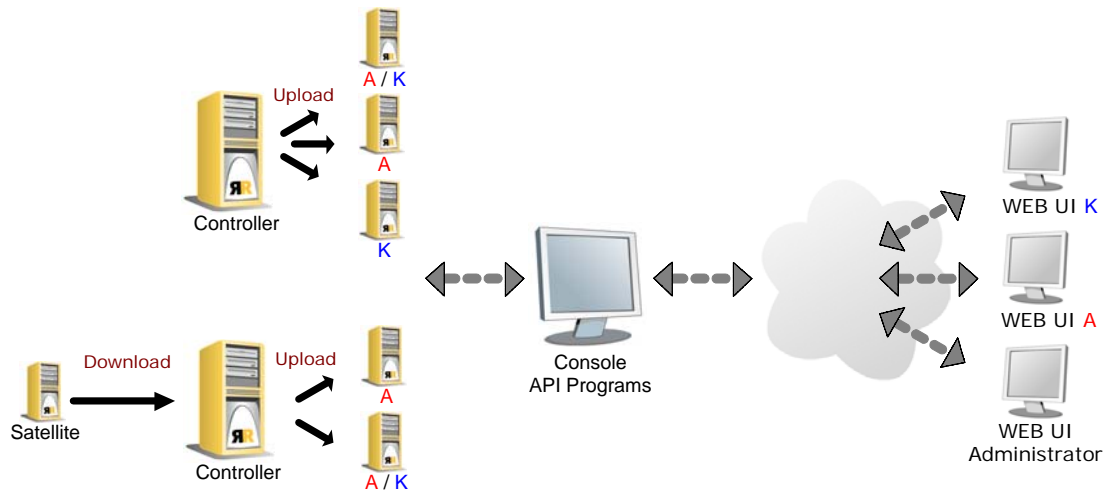
The RepliWeb Deployment Suit Application Programming Interface (API) allows applications written by the user to access RDS’ robust functionality. User applications can submit, monitor, and control replication jobs throughout their lifecycle.

It should be understood that conceptually it is not different from the CLI approach. Still, a central machine, not related to the replicating machines, is polling Controllers for their jobs information. The difference is in the processing abilities in a program using the API as opposed to a script using the CLI.

The classic example of implementing this is using the API in a program that is run by a web server. That program polls the Controllers that it should poll and generates output that is relevant for the user that should monitor the specific process. Having the web server run this program continuously, creates a near real time monitoring of a logical group of replications. Being run from a web server, makes it possible to run several, different programs, each monitoring and controlling a different set of replications.

Another example is the simplicity of implementing control functions like aborting a distribution process that is currently running on multiple Controllers. Taking the above approach, the real topology can be hidden from. The user can simply instruct to “abort”

and the program will be run will use the API to access all relevant Controllers and abort all relevant processes.



Using the API on the monitoring node under a web server, expands the 3-tier architecture into 4-tiers. This expansion is done without the need to install any RDS component on the monitoring nodes used by the end user (only a browser is required) and at the same time giving the end user only the exact functionality that is required to perform a specific, logical, task. The connection between the monitoring end user and the monitoring node can be based on http or https is a secured and authenticated connection is required.

For a detailed explanation of the RDS API please refer to the RDS API Reference Guide located in you installation directory and in the RepliWeb Reference Library.

RDS installation kit provides complete programs that use the standard API to build a full Console Web based interface.

## 4. Appendix I – RDS CLI and XML output examples

### 5.1. Getting job information from a Controller

The following is an example for getting the list of jobs and basic information on each job from a single server (ServerA). The list of jobs returned would be the list of jobs running on behalf of UserA.

```
> rds show -query=detailed -controller=ServerA  
-controller_user=UserA -controller_password=Password
```

This will give an output of all jobs on the Controller ServerA running under UserA.

### 5.2. Sending the output to a file

Sending the output to a specific file (on the monitoring node) is done by adding to the above command the qualifier:

```
-output=file_name
```

### 5.3. Appending the output to the same file

Appending the output of multiple commands to the same file is done by adding the qualifier:

```
-output_mode=append
```

### 5.4. Generating XML output

When instead of standard terminal output, XML output is required so that the output can be parsed by another application or process, the following qualifier should be added:

```
-output_style=tagged
```

## 5.5. Getting detailed information for every job

By default the `rds show` command will generate only basic information about each job on the Controller. Generating detailed information is done by adding:

```
-query=detailed
```

### Example

The following will generate detailed information in XML format about the jobs running on ServerA on behalf of UserA.

```
> rds show -query=detailed -controller=ServerA
-controller_user=UserA -controller_password=Password
-output_style=tagged -output=c:\output\serverA.xml
```

## 5.6. Tagged output example

The tagged output gives each qualifier and descriptive an open and close tag the output from a single Controller would look like:

```
<rds_output>
  <rds_job>
    <id>Job_id</id>
    <state>
      Submitted/Running/Hold/Recovering/Failed/Scheduled</state>
    <direction>Up/Down</direction>
    <logic>Mirror/Backup/Purge</logic>
    <stage></stage>
    <name>Job Name</name>
    <source_dir>Source path</source_dir>
    <destination_dir>Target path</destination_dir>
    <satellite>Satellite Server</satellite>
    <description>Job Description</description>
    <file_specs>As defined in Job</file_specs>
    <attempts>No of attempts if recovering</attempts>
    <files_copied>Number of files copied</files_copied>
    <files_to_be_copied>Number of files to copy
  </files_to_be_copied>
    <streams>Number of streams</streams>
    <bytes_transferred>Number of bytes transferred
  </bytes_transferred>
    <files_failed></files_failed>
    <stream> Each stream then has details about it's own
      progress
      <stream_number>1</stream_number>
      <stream_state></stream_state>
      <stream_files_copied></stream_files_copied>
      <stream_files_to_be_copied></stream_files_to_be_copied
    >
  </stream>
    <files_deleted>Number of files deleted</files_deleted>
    <differential_transfer></differential_transfer>
```

```

        <bandwidth>Expression</bandwidth>
        <user>User</user>
        <domain>Domain<\domain>
        <submit_time>Time</submit_time>
        <end_time>Time</end_time>
        <schedule_time></schedule_time>
        <schedule_type></schedule_type>
    </rds_job>
    <message>
        <type>Success</type>
        <text>RDSAPI-S-SHS, successfully displayed N jobs </text>
    </message>
</rds_output>

```

## 5.7. Polling multiple Controllers

If you now have multiple Controllers and you want an XML output from all jobs you need to create a batch file job with an **rds show** command per Controller. If you want to combine the output into a single XML file, the batch file must add a start tag and an end tag to the xml output

A typical script would look like this:

```

> c:\temp\start.tag > c:\temp\output.xml

>rds show -controller=ServerA -controller_user=UserA
-controller_domain=Domain -query=detailed
-controller_password=Password -output_style=tagged
-output=c:\temp\output.xml -output_mode=append

>rds show -controller=ServerB -controller_user=UserB
-controller_domain=Domain -query=detailed
-controller_password=Password -output_style=tagged
-output=c:\temp\output.xml -output_mode=append

> c:\temp\end.tag >> c:\temp\output.xml

```

With `start.tag` being a file that contains:

```
<Multiple Controller Output>
```

and `end.tag` being a file that contains:

```
</Multiple Center Output>
```

This gives an the combined output in the file `c:\temp\output.xml` viewed in a browser as below:

```

-<Multiple Controller Output>
+ <rds_output>
+ <rds_output>
</Multiple Controller Output>

```

Each `<rds_output>` corresponding to each Controller that is in the batch job.

## 5.8. Narrowing it to just the running jobs

If you want the output to contain information just about the currently running jobs, you need to add to each `rds show` command the qualifier:

```
-state=running
```

For a detailed description of the CLI please refer to the [RDS User Guide](#).

###

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