

R-1 Building Centralized Monitoring and Control

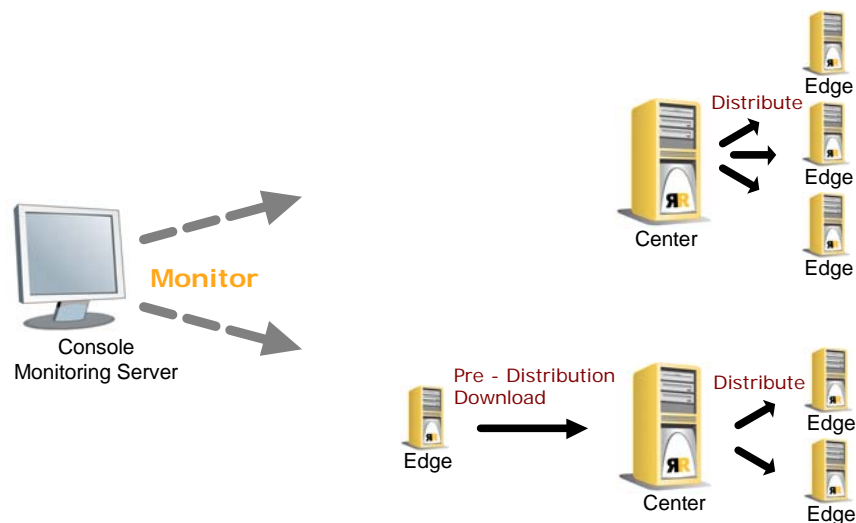
1. Overview

This document explains the concept and differing options for the monitoring and control of R-1 distribution over your network.

The very basic nature of R-1 is a 3-tier architecture where a detached Console software component defines and monitors a distribution process that takes place between a two or more machines (a Center and Edges). That architecture is preserved when doing central monitoring in the sense that the distribution information is retrieved (by Console, Command Line, or API) from the Center. To present a centralized picture of multiple distributions taking place on multiple Centers, multiple Centers are accessed.

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

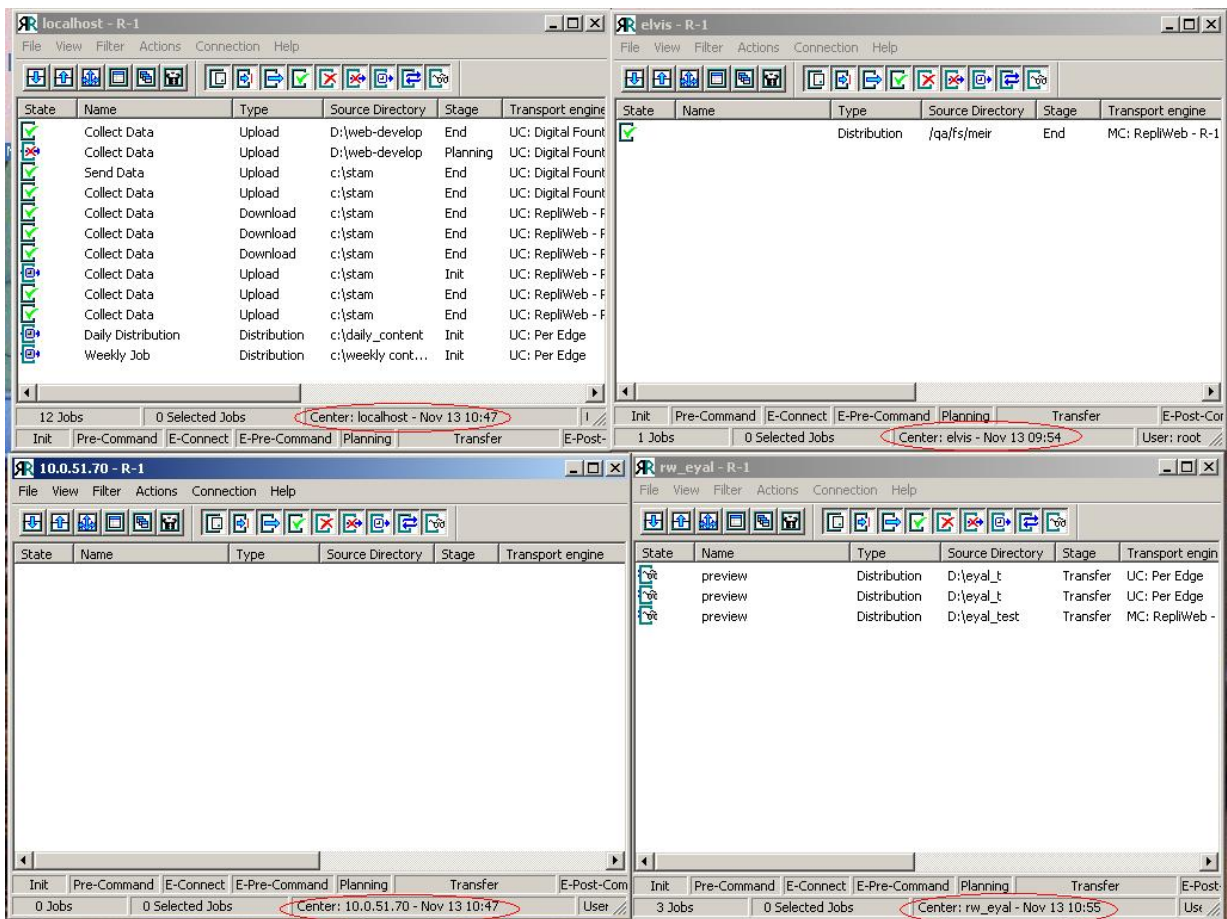
The following diagram describes the desired topology where distribution takes place between a variety of servers, and the monitoring node is not a part of the distribution topology. The key point, as mentioned before is that the monitoring node needs to communicate (in a variety of ways that will be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



2. Using Multiple Consoles

This approach is recommended for small topographies or classic star configuration. Small topographies are in the sense of the numbers of Centers involved in the distribution, 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 multiple Consoles open, each connected to a different Center.

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 Centers are involved. This is completely not practical when more Centers are involved.



3. Polling Multiple Centers using R-1 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 distributions taking place all over the network is contained in the Centers of the network. In order to get a centralized picture, the information from all these Centers should be gathered and consolidated into one central picture. This is done by building a simple batch file that connects in turn to each Center and writes the job information from that Center 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 Center 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 Centers 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.

4. 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 distribution processes. In real networks, there can be multiple distribution processes running in different or common parts of the network.

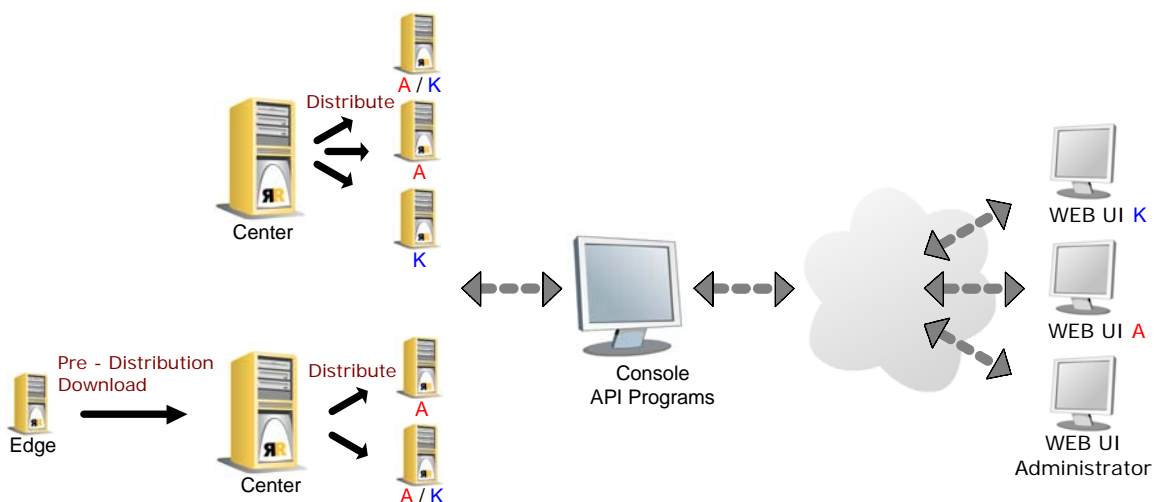
They can be serving different needs like content deployment, or collection of data for disaster recovery. Even though they can be running on the same machines they are not connected, logically, and need to be monitored and controlled by different people. The specific monitoring can include the need to see different and customized information. In these cases, the approach should be to use the R-1 Application Programming Interface (API) to poll the right information from the right Center and build the proper information display.

The R-1 Application Programming Interface (API) allows applications written by the user to access R-1's robust functionality. User applications can submit, monitor, and control distribution 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 distributing machines, is polling Centers 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 Centers 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 distributions. Being run from a Web server, makes it possible to run several, different programs, each monitoring and controlling a different set of distributions.

Another example is the simplicity of implementing control functions like aborting a distribution process that is currently running on multiple Centers. Taking the above approach, the real topology can be hidden from users. The user can simply instruct to "abort" and the program will be run will use the API to access all relevant Centers 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 R-1 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 if a secured and authenticated connection is required.

For a detailed explanation of the R-1 API please refer to the R-1 API Reference Guide located in you installation directory and in the RepliWeb Reference Library.

R-1 installation kit provides complete programs that use the standard API to build a full Console Web based interface.

5. CLI and XML Output Examples

5.1. Getting job information from a Center

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.

```
> r1 show -query=detailed -Center=ServerA  
-Center_user=UserA -Center_password=Password
```

This will give an output of all jobs on the Center 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 `r1 show` command will generate only basic information about each job on the Center. 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.

```
> r1 show -query=detailed -Center=ServerA
-Center_user=UserA -Center_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 Center would look like:

```
<R-1_output>
  <R-1_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>
    <Edge>Edge Server</Edge>
    <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>
</R-1_job>
<message>
    <type>Success</type>
    <text>R-1API-S-SHS, successfully displayed N jobs </text>
</message>
</R-1_output>

```

5.7. Polling multiple Centers

If you now have multiple Centers and you want an XML output from all jobs you need to create a batch file job with an **r1 show** command per Center. 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

>r1 show -Center=ServerA -Center_user=UserA
-Center_domain=Domain -query=detailed
-Center_password=Password -output_style=tagged
-output=c:\temp\output.xml -output_mode=append

>r1 show -Center=ServerB -Center_user=UserB
-Center_domain=Domain -query=detailed
-Center_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 Center 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 Center Output>  
+ <r1_output>  
+ <r1_output>  
</Multiple Center Output>
```

Each `<r1_output>` corresponding to each Center 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 `r1 show` command the qualifier:

```
-state=running
```

For a detailed description of the CLI please refer to the R-1 User Guide.

###

For any additional information, contact us at support.repliweb.com