Datto RMM

Integrations Whitepaper for Third Parties

February 2025 · Build 12
SGL

Contents

Welcome to Datto RMM

Thank you for your interest in the Datto Remote Monitoring and Management (RMM) platform!
Since 2006, Datto RMM – first as CentraStage, then as Autotask Endpoint Management (AEM) – has been the go-to solution for managing and monitoring endpoints from both high- and low-level views. Datto RMM permits effortless, sweeping automation and granular configuration as required by the user. We are eager to work with third-party vendors to increase the scope and capabilities of the product, and we look forward to working with you throughout the process to create an integration that both sides can take pride in.

What is Datto RMM?

Datto RMM is a cloud-based endpoint management solution operating on the Software-as-a-Service (SaaS) model. Endpoints are managed either by installing the Datto RMM software (the "Agent"), or via SNMP requests sent by a nominated network device in cases where software installation is not supported (for example, monitoring network devices). ESXi integration works under the same principle. The Agent is tested on Microsoft Windows (10/Server 2012+), macOS (last two versions), and Linux (various) operating systems.

Management operations are conducted either via the Web Portal or via the Agent depending on the granularity of the action. Operations are handled via Amazon Web Services (AWS), to which devices connect and from which management commands are arbitrated. More information on backend and infrastructure.

Managed devices regularly send information back regarding their running state to the platform, which catalogues the data in an easily digestible format. Arbitrary commands can be sent directly to devices via "Components" – scripts – to perform tasks like software installation, and intelligent monitoring built into the product can advise on devices' antivirus and patch management status automatically following implementation.

What can Datto RMM be used for?

Through the Agent, Datto RMM provides a complete top-down overview of any device added to it.

Usage examples include:

- Using the data within each device's software audit database to ascertain which devices are missing a specific type (or version) of a software program, and to then place that information in a dynamic filter against which a job installing the software silently (or a report cataloguing the data) can be targeted.
- Monitoring (based on real-time monitoring data) various system faculties (disk usage, service health, anti-malware suite status etc.) and alerting the administrator if all is not as it should be.
- Funnelling and synchronizing alerts to tickets within a PSA system, from which such tasks can be delegated and managed in greater detail.
- Connecting to an individual device and performing maintenance online, either in full view of the end user or with local viewing disabled.
- Production of detailed reports highlighting important data, or presented in summary form to demonstrate status, device health, work done, etc.
- Much more!

Datto RMM also offers an API which can be used to return information to third parties outside of the purview of the Agent in cases where Agent communication via Components is not a practicable solution.

Basics of integration for vendors

Datto RMM is a well-established RMM solution that serves over eight million endpoints. Since 2006 our customers have trusted us to manage and catalogue their devices for them, with more being added every day.

By integrating with Datto RMM, you can put forth a unified solution by presenting your product to customers through an interface they are already familiar with; furthermore, prospective customers looking for a new solution may treat your integration with their existing RMM as a distinguishing virtue among candidates.

Integrating within the Component Model (Recommended)

While an integration may take many forms depending on the service, protocol biases strongly toward integrating via the Datto RMM Component engine as opposed to developing custom features for integration partners into the Datto RMM interface. (See the next chapter for an overview on Datto RMM Components.) Integrations following this pattern typically consist of three distinct Components with defined roles:

- **Deployment**
  - Using a Component, software can be deployed silently to devices almost instantly.
    The Component engine uses command line functionality present in the system itself, so anything that can be accomplished via a Batch or PowerShell prompt can be done as part of a Component, enabling bulk actions and automation across vast collections of devices.
  - Provided such actions are theoretically possible, a deployment Component can handle both the task of deploying the necessary software as well as registering it to ensure that critical licensing and configuration data are supplied.
  - Updates to existing installations can easily be provided via the same Component engine, permitting simple lift-out/lift-in upgrades in scenarios where such operations are possible within the software.
- **Monitoring**
  - Using the Component engine, it is possible to report back more detailed information using script-based monitors which can perform checks against the system and return their findings as alerts.
- **Management**
  - While Datto RMM does not poise itself as a replacement for a vendor's native product management interface, in cases where a program can be sent management commands (for example, to update or run a scan via the Registry or command line), such actions can be managed via Components.

Integrating outside of the Component Model

API

If your product is a cloud service and does not provide software *per se*, Components may be a poor fit. Datto RMM has an oAuth 2 REST API which can be used to remotely pull information and send it across to third parties; please consult the relevant section for more information on how to integrate with it.

UAV

If your product is an antivirus suite, you can apply for inclusion in Datto RMM's Universal Antivirus Audit (UAV). Please consult the relevant section for information on this service and how to integrate with it.

On Components

Overview

Executing arbitrary code on an endpoint via Datto RMM – for example, to install software, or to run a script – is handled primarily via the delivery and execution of code directly on the endpoint.

Datto RMM accomplishes this by distributing and executing “Components” on endpoints, with a “Component” being defined as a package containing:

- **Metadata** (a description of the Component and an icon for the RMM Web interface).
- **Code** to be executed in script form – generally Batch, PowerShell or Shell depending on the target.
- (Optional) Other **attachments** required for functionality, such as a software installer.

Whilst all Datto RMM users can make their own internal Components that are visible only within their respective user accounts, there is a public repository (the “ComStore”) from which customers can download pre-made Components made available by Datto RMM staff. Integration Components are to be submitted directly to your Datto RMM integrations delegate, who will verify them before uploading or updating them on the ComStore for public availability. More on Components.

Datto strongly recommends working any RMM integration into this Component-based format as opposed to custom Web Interface development due to the seamless nature in which Components can be uploaded and updated for end users within the ComStore. A Component update can be processed, delivered, and distributed to customers in a fraction of the time required to develop and quality test product-level code adjustments. Speak with your Datto RMM integrations delegate for more information on producing Components for RMM. (As a reminder, Datto RMM has an API for cases where the Components model is not appropriate.)

Production

Making a Component

A Component is made by writing a non-interactive script (the most viable languages are Batch or PowerShell for Windows and Shell for macOS and Linux) and entering it into the script window.
Any text displayed in the console (either with ‘`echo`’, ‘`write-host`’ or ‘`printf`’) will display as part of the script’s standard output (“StdOut”), which will become visible after running the Component as a Job on an endpoint. Please see the documentation on Making a Component Script and Making a Component Monitor.

Datto RMM is not and does not contain a script interpreter; the script is sent directly to the endpoint along with the other contents of the Component, and it is the endpoint that executes it. As such, care must be paid towards compatibility; when writing scripts in PowerShell, ensure you are using commands that are valid for your target device’s OS (Windows 7 ships with PowerShell 2, for example).

A Job ‘success’ or ‘failure’ depends on the exit codes of the applications that run as part of the script scope and within the script itself. Provided everything exits with a code of 0 (‘no error’) the job will succeed. It may be necessary – for example, in case of incompatibilities – to force exiting with a code 1 in some cases.

User Variables

A ‘User variable’ is a variable that is used within the script but defined outside of it, being set instead at runtime by the user running the Component. This is a great way to populate customer-specific information, such as serial numbers as well as Booleans to control options; ‘Selection’-type variables can also be used to provide user-friendly options (“Perform Scan”, “Uninstall and Reinstall”) to flag certain actions within the script itself.

Furnishing User Variables at the Site Level

In addition to supplying values for User variables at runtime, values for User variables can be pre-furnished in the Datto RMM interface at the Site level using the Site Variables feature. From the Site level Settings page, an administrator can provide a default value for any variables they choose.

If Site- or Account-level variables are to be used, they must not be defined at the Component-level. Defining the variable at the Component-level *and* the Site- or Account-level will cause the Component-level value to be taken in all circumstances, even when it is empty.
Site- and Account-level variables can be set via the API.

This opens the door to mass deployment across multiple Sites using different serial numbers; each Site will need to have its own value defined for whatever variable name has been chosen within the script. Site variables can be updated at the Account level by downloading, editing, and then re-uploading a CSV file containing the intended values of the variable for all Sites. More information.

Agent Variables

Some variables are sent alongside each Component deployment. These variables can be declared within the script without being set beforehand in order to gather information about the partner running the script.
The information includes:

- Datto RMM Platform
- Site Name
- Account Name

More information on Agent variables.

Caveats

There are a number of points to be aware of when using Components:

- Quick Jobs, through which the majority of Components are run, execute at the most highly-privileged user account context. On Windows this is NT AUTHORITY\SYSTEM; on macOS and Linux it is `root`.
- Due to this privileged execution, jobs are not visible to any logged-in user; as such, all Components must run silently and must not prompt the user for interaction as the user will never be able to see the installation. Any interactivity will cause the script to hang in memory until it times out and is terminated.
- Datto RMM does not support deployment of APPX-based Windows Apps.
- Components should **never** reboot automatically, but should instead inform the administrator that the device will need to be rebooted before using the software. (Rebooting via a User variable option is acceptable.)
- If a program starts the moment it is installed, it may do so as part of the script scope, causing the script to hang until the program has exited. In this case, the program may need to be terminated by the script.

If you are ever unsure about how (or how *best)* to do something via a script, reach out to your integrations delegate – they may know of a way to accomplish what you need.

Publishing a Component

The responsibility is on the software vendor to write and support their own Datto RMM Components.
Your Datto RMM integrations delegate will make themselves available to help troubleshoot and to provide guidance on Components where necessary, but the task of production, testing, and maintenance of the Component is incumbent on the vendor. Support queries with integration Components will be delegated to the vendor via a nominated member of their staff, so please ensure you have a person or department on-hand to accept support queries in this instance.

Upon finishing a Component for use with Datto RMM – either for the first time or as an update to an existing Component – the partner will be required to speak with the Datto RMM integrations team in order to get the Component submitted through internal testing and placed in the ComStore for customers to download.

A Component submission may not be accepted immediately, and may require tweaking in order to meet the standards for ComStore Components. For reference, beyond functionality, these are:

- Icon must be a transparent PNG of 48x48 dimensions
- Supported OS must be listed at the end of the Component name – "[WIN]", "[LIN]" or "[MAC]"
- Component must function on at least Windows 10/Server 2012, and must quit gracefully with an explanation if the script is run on hardware known to be incompatible
- Text in the error stream (StdErr) must be kept to an absolute minimum (logging data is OK)
- Script must not reboot devices automatically

Getting data back into the Portal with User-Defined Fields

We have already discussed how Components can be used to run scripts on devices at the most highly-privileged user levels, with the upcoming sections discussing how this functionality can be leveraged to deploy, configure and monitor software; there is, however, another provision which can be used to gather data from the endpoint in such a way that it becomes part of the Datto RMM device record for a 'round-trip' experience.
This functionality can be found in the form of "User-Defined Fields" (UDFs), where the 'user' is the person using the Datto RMM product to administer endpoints.

Any Managed device with an Agent on it will support up to thirty of these fields – a string numbered 1-30 capable of holding up to 255 characters of text – which can be altered either within the Web Portal or directly on the device by way of the Registry (for Windows devices) or XML files (for Unix devices).
UDF Data placed on an endpoint is synchronised directly with the platform instantly; UDF data can also be pulled back from the Web Portal into a Component script by using special reserved variables.

Possible use cases for this functionality are:

- Calculating a device-specific hash using data unique to the endpoint and then placing it into a UDF for the purposes of reporting or inventory management (Script > Agent > Platform)
- Creating device-specific hyperlinks to third-party services and placing them into UDFs so they can be followed from the Web Portal (Script > Agent > Platform)
- Assigning devices to arbitrary categories that can be pulled from the product during a script run in order to instruct a script which behaviour model it should follow (Platform > Agent)

Find out more about User-Defined Fields in the Datto RMM Online Help documentation.

Deploying Software via a Component

Overview

“Deployment” within the context of this document refers strictly to the act of deploying the vendor’s software onto an endpoint using the Datto RMM software and a Component package. For information on integrating outside of the Component model, refer to Integrations outside of the Component or API Models.

Deploying software via Component is an exercise in consideration; while the process may appear simple, there are many aspects to consider that may complicate things. In a nutshell, the goal of any successful software deployment via Component is to get the vendor’s software installed onto an endpoint in a ready-to-use state, and in a manner that is as transparent to the end-user as possible. We reiterate here that installations are always hidden - the user cannot interact with the installer in any way because they cannot *see* it. As such, all operations must be non-interactive.

Such an approach begs many questions:

- How is my software **installed**?
    - Is it via an MSI, or does it use a third-party installation system - in which case, is there a silent flag provisioned within the installer framework?
    - Is the installation typically a two-step process, with the first step being extraction of the files?
    - Is my installer generic or produced ad-hoc on a per-deployment basis? (See “Need to Include”)
- What **requirements** befit a successful installation into a ready-to-use state?
    - How is my software licensed/registered? Can this be achieved without user input?
    - Does a successful installation require the user to reboot their computer? Can this be overridden?
- What **data** do I need to permit the user to pass along to the installation process?
    - Does my installer require a serial key or a language selection?
    - Does my installer support command line flags to process this data, or must the installation script perform these steps post-installation in the Registry (for example)?
- What **environments** does my software support?
    - If my software does not support a certain OS or requires a certain dependency, does my installer script catch installations outside of these parameters with a useful error?
    - Does my software require a specific OS architecture? Does the script catch devices outside of this?
- What do I **need to include** with my installer script?
    - Are there any command line tools the script uses to facilitate the installation – cURL, for example?
    - How large is my installer package? Can this data be compressed?
    - If activation is handled via keyfiles, can these be generated by the installer script using information supplied by the user at run-time?
    - If the script downloads anything, are those downloaded files verified before execution?
- How does the installer **finish**?
    - Does the software immediately launch itself post-installation?
      If so, it may need to be terminated so as to prevent it from opening in the wrong user context.

Asking these questions as the Component is being produced will help ensure its quality and reliability. Please remember that as the Vendor, supporting these Components is incumbent on you, so it is wise to start off with a solid foundation on top of which future improvements and/or features can be built.

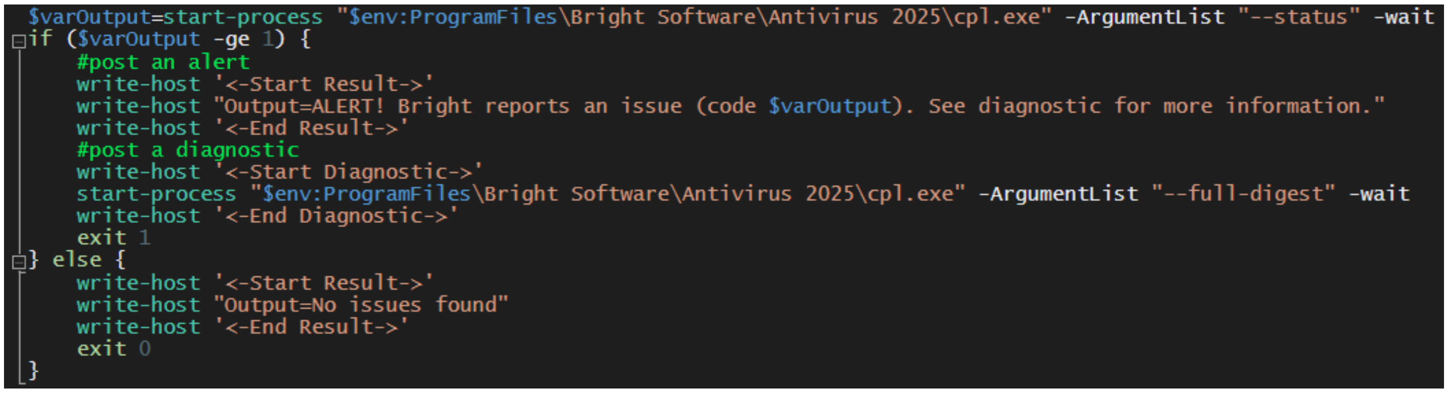
Monitoring via Component

Using the same Component engine as would be employed for tasks of Deployment, Component Monitors can be written to perform regular checks against a device or to perform maintenance. These take the same form as Application- or Script-type Components, but with some notable differences:

- Monitors can sound **Alerts** which can contain **Diagnostics** – more information below.
- Files cannot be attached to Component Monitors.
  If your monitoring solution requires the presence of a file which may not be present on the system, the Monitor script should be written such that these files are downloaded first (and then verified).
- Monitors are stateless. No Monitor run has any knowledge of the run immediately preceding it.
  This may necessitate usage of outside services (such as the Registry) to store longer-term data in order to enable a Monitor to report on cumulative issues, as a Monitor is not capable of asking its previous run how many errors (for example) it encountered.
- Monitors can run concurrently. If the user has your Monitor set to run once a minute, it will be run once a minute, even if the previous run has not yet concluded. This may also necessitate storage of some data outside of the monitor such that subsequent runs can exit immediately if they notice that the previous Monitor run is still ongoing at time of execution.

A Basic Example

A Monitor Component might consult an executable found on the local device and sound an alert if its data is greater than zero. It can then sound an **Alert** notifying the user with a summary of the issue and also post a **Diagnostic** giving more information at the same time. These Alerts can be forwarded to any ticketing system the user has connected to Datto RMM, such as Autotask PSA (another Kaseya product).



```
$varOutput=start-process "$env:ProgramFiles\Bright Software\Antivirus 2025\cpl.exe" -ArgumentList "--status" -wait
if ($varOutput -ge 1) {
    #post an alert
    write-host '<-Start Result->'
    write-host "Output=ALERT! Bright reports an issue (code $varOutput). See diagnostic for more information."
    write-host '<-End Result->'
    #post a diagnostic
    write-host '<-Start Diagnostic->'
    start-process "$env:ProgramFiles\Bright Software\Antivirus 2025\cpl.exe" -ArgumentList "--full-digest" -wait
    write-host '<-End Diagnostic->'
    exit 1
} else {
    write-host '<-Start Result->'
    write-host "Output=No issues found"
    write-host '<-End Result->'
    exit 0
}
```

The Agent automatically catches output formatted in a specific manner alongside the exit code and can sound Alerts or exit with none depending on the makeup of the script and the state of the device running the script. Anything that can be formatted into a PowerShell (or Bash, or Batch) `if` block can be used with this logic.

Alerts are de-duplicative; a Monitor cannot fire a subsequent Alert until the previous one has been resolved, either manually by the user or automatically once the conditions to trigger the alert stop occurring.
If you need Alerts to function independently of each other, you must write entries to the Event Log and then instruct users to manually monitor for instances of your events appearing in that log.

The API

Datto RMM provides an Application Programming Interface (API) to enable programmatic access to information and operations available in the Datto RMM Web Portal. API access is implemented via a REST interface, and the available requests are documented online.
More information on the Datto RMM API.
Available requests.

As an integration partner, you receive access to a Datto RMM account which can be accessed via the API. The API is set up using oAuth 2 such that only the details of the account authorised can be accessed; there is no upper-level API to which only special access is granted which grants additional features or control over the accounts of others.

Some features (like, for example, creation of Components) are not available via the API and there is no intention to surface them. This is due to concerns over security.

The following features of the API may be of particular interest to integration partners specifically:

- Displaying all Sites and Devices in a customer's account
  Display all Datto RMM Sites in order to compare them against how the customer has configured your own platform in order to accomplish mapping between the two systems
- Setting and checking Site- and Global-level variables
  Set a licence key at the Site-level such that all software deployments for that site using your ComStore Component automatically use the correct licence information for that Site
- Retrieve Stdout/Stderr from Job runs
  Check to ensure that a deployment job finished correctly
- Check a device's UDFs
  Configure a Component Monitor to write its status to a UDF and check its contents automatically

Please remember that **Datto RMM is device-centric first.** If specific information is not accessible via the API, it is because the assumption would be that the Agent and a script – if not the vendor's own software – would access the device and return that information. Not all device data is surfaced by the API and the most fully-featured integrations make use of a combination of it and dedicated software.
Alerts cannot be "pushed" into the API for a device. In-keeping with the theme of device-centricity, a device can only make alerts for itself.

Your integrations delegate will be on-hand to answer any questions you may have about using the API.

Universal Antivirus Audit (UAV)

Datto RMM can intelligently detect the presence of antivirus software, its running status, and whether it is up-to-date using information available to the Agent as opposed to via a Component. This information is then synched directly to the device's antivirus status fields (it can also be explanted there using a JSON file).
Speak with your integrations delegate about getting your antivirus's information into Datto RMM's UAV.

Management via Component

Management tasks should generally be conducted via the integration partner's own web portal; however, using Components it is possible to send commands to antivirus solutions on the endpoint level provided the software is capable of processing instructions provided to it silently from a manner reachable by a Component. Your software may react to a registry value for "perform a scan" being changed, for example, or to a certain program being run with a command line-flag (say, `runscan` or `status`).

A novel way to encompass multiple commands in a single Component (to reduce the need for single-use Components labelled "Search for Updates", "Run a Scan", "Uninstall" etc.) is to use a User variable of the type "Selection", which can then be mapped to one of a list of options. A management Component's script may, for example, invoke a "switch" clause to check whether the value "$action" is given as "uninstall", "scan", "update", etc., with the value of $action being set at run-time by the user from a dropdown list of potential actions as stipulated by the Component author.

Integrations outside of the Component or API Models

While Datto strongly recommends that all integration partners work on the Component- or API-level to achieve their integration, we appreciate that this approach does not cater to all use cases. Integration with a ticketing system, for example, ill-befits a Component-based approach. If you believe that your integration can only be accomplished by way of alterations to the Datto RMM product, please speak to your integrations delegate.

Summary of points

- Datto RMM is open to integrating with third-party software and appreciates your interest.
- The majority of integrations can and should be accomplished using the Component model, where tasks like deployment, management, and monitoring of your software via ours is accomplished by pushing arbitrary code to endpoints and monitoring the results.
- Datto RMM also offers an API which can be leveraged in addition to, or instead of, the Component model in cases where it fits more appropriately.
- Components must be written and tested by the vendor.
- The vendor must nominate an employee who will be delegated support tickets in the event of an RMM issue with a Component written by the vendor.
- Non-Component-based integration applications should be made individually in writing to the RMM integrations team.

Example Case Study: Bright Antivirus 20XX

Overview

Bright Software Co., Ltd., a fictional software developer in Osaka, produce Bright Antivirus. An executive decision has been made to work with various technology services providers with the goal of making it easier for MSPs to roll out Bright Antivirus using fewer interfaces, so simplifying the deployment process and facilitating deployment at-scale; a second requirement is to permit monitoring of the Bright Antivirus solution such that alerts and tasks can similarly be managed from the same interface used to deploy the software.

The steps outlined by Bright for a complete integration with Datto RMM specifically are:

1. Deployment of Bright Antivirus software to Windows and macOS Endpoints
    - Application of licence information during the installation process
    - Device-level support for linkage *from* Datto RMM *to* Bright's own console
2. Support *within* Datto RMM for at-a-glance Antivirus status readout
3. Monitor alerts within Datto RMM for Bright Antivirus alerts demanding client attention
4. Ability to push commands to update and run full scans from the Datto RMM interface
5. Oversight of Component popularity from Datto RMM to gauge market response

1: Deployment

Operating System

Whilst it is technically possible to produce Components that work on both Windows and macOS, Bright decided instead to produce two distinct Components for Windows and macOS respectively in order to gain more granular insight into usage statistics after the integration was public. Accordingly, Bright's engineers produced a Windows script in PowerShell. The macOS script was written in Bash, but a small AppleScript file to perform additional maintenance was also attached for invocation within the script using the OSAScript utility.

Licence Information

Bright's licence keys roughly correspond to Datto RMM's concept of *Sites*, such that a licence key would be used to deploy to a single partner in a single geographical location. The scripts were therefore designed with User variables; this decision was taken over the use of Site-level variables to permit users to perform deployments on single devices with unique keys instead of forcing a direct Site-to-Key correlation.

On Windows devices, the licence key is a sixteen-character string that must be placed into the Registry; the installation script thus creates a Registry value in the appropriate 32- or 64-bit location of the HKLM hive. (Components run via Datto RMM generally execute as NT AUTHORITY\SYSTEM; scripts run at this level will map HKCU data as the LocalSystem user as opposed to as the logged-in user, which is generally unuseful).

The macOS installation routine handles registration in its AppleScript file; OSAScript is invoked referring to the SCPT file attached to the Component, with the licence key as an invocation parameter.

Cross-platform Linkage/Mapping

Having been informed of the potential of User-Defined Fields, Bright wrote a small command-line utility containing the same hashing algorithm used within the Antivirus software to calculate a device's unique ID within its web portal. This data consists of the MAC address of the first active NIC of the device and its motherboard serial number. The command-line tool, invoked via the installer script, generates the hash. **Other potential hashing values include the Site Name or the Partner's Account Name.**

Bright Antivirus's URL schema looks like https://partner.bright-av.web/console/device?id={ID}.
The final part of the installer script places a Registry value (or a value in values.xml for macOS) mapping a UDF, the number for which set as a User Variable, to the value of the above URL, with "{ID}" replaced with the device UID as calculated by the tool.
In this manner, the user can click the link in the UDF to be sent to Bright's own console page for the device.

2: Status Readout

Bright Antivirus reports to Windows' SecurityCenter2 WMI repository, meaning Datto RMM is able to ascertain automatically its name and running state *on Workstations*; however, this leaves Windows Server users unable to gauge antivirus health.
Accordingly, Bright sent Datto RMM the following information regarding status checks:

- To check whether the product is **running**, see if BvAntiSrv.exe is running in memory.
- To check the last time the product performed an **update**, run the following command:
    - *(Program Files x86)/Bright Antivirus/BvCmd.exe --status*
    - The response will be a date in Unix Epoch format; any date older than seven days should be considered "out-of-date".

This information is worked into Datto RMM's monitoring Agent and gathered automatically by the software.

3: Monitor Alerts

Bright Antivirus writes to the Event Log with code 26108 when an alert is triggered. Accordingly, Bright sent Datto a Monitoring Policy in PCY format with Event Log monitors configured within; Upon receipt, Datto placed this Monitoring Policy into the "Best Practice" Policies section.

Other methods of gathering and reporting on data by artefact are command-line tools, Registry traces, file presence, etc.; these all must be gathered by a script, however.
How to write a Monitor Script for Datto RMM.

Datto RMM is device-centric; this means that there is no intrinsic support for "pushing" an alert from (in this case) Bright's service directly into Datto RMM's for the device listing there. **The logic should begin on the endpoint** to detect the presence of an alerting criterion and respond.

4: Sending Commands to the Software

As noted in 2), Bright Antivirus ships with a command-line binary (*BvCmd.exe*) which is used in that instance to gain insight on the date of the last engine update; using the "--Update", "--QuickScan" or "--FullScan" commands will similarly cause the software to perform an update or a scan respectively.

Before BvCmd's introduction, a prior method of sending commands to the software via the local endpoint was by placing data into a Registry value supervised by the software; changing the data for the Update, Quick or Full Scan values respectively from 0 to 1 would cause the software to react accordingly once aware.

Either of these approaches can easily be made into a script Component which could itself be distributed in the Datto RMM ComStore as a "tasks" Component.

5: Community Response to Component

While integration partners do not have direct access to the Datto RMM ComStore backend, arrangements can be made with Datto staff to gather metrics on a Component's popularity on a monthly (or less frequent) basis. Ask your integration delegate if you would like to receive updates on how many downloads your Component/s have had within Datto RMM.

Please note that the only metric available at the time of writing regards a Component's total download count; a "download" is performed at the account-level and cannot be used to ascertain how many *devices* or Sites a Component has been run on. Update actions for already-downloaded Components also do not increment the counter; treat it as a count of "total # accounts using the integration".

Datto staff are unable to provide lists of individual partners using an integration.

Further reading

Get in touch with the Datto RMM Integrations team

E-mail the Datto RMM integrations team at rmm-integrations@datto.com.

Datto RMM Online Help

Find documentation on how to use Datto RMM effectively in our Online Help: https://rmm.datto.com.