



18 min read

Microsoft Sentinel

Demystifying Data Collection Rules and Transformations



Robbe Van den Daele

May 21, 2023 • 18 min read

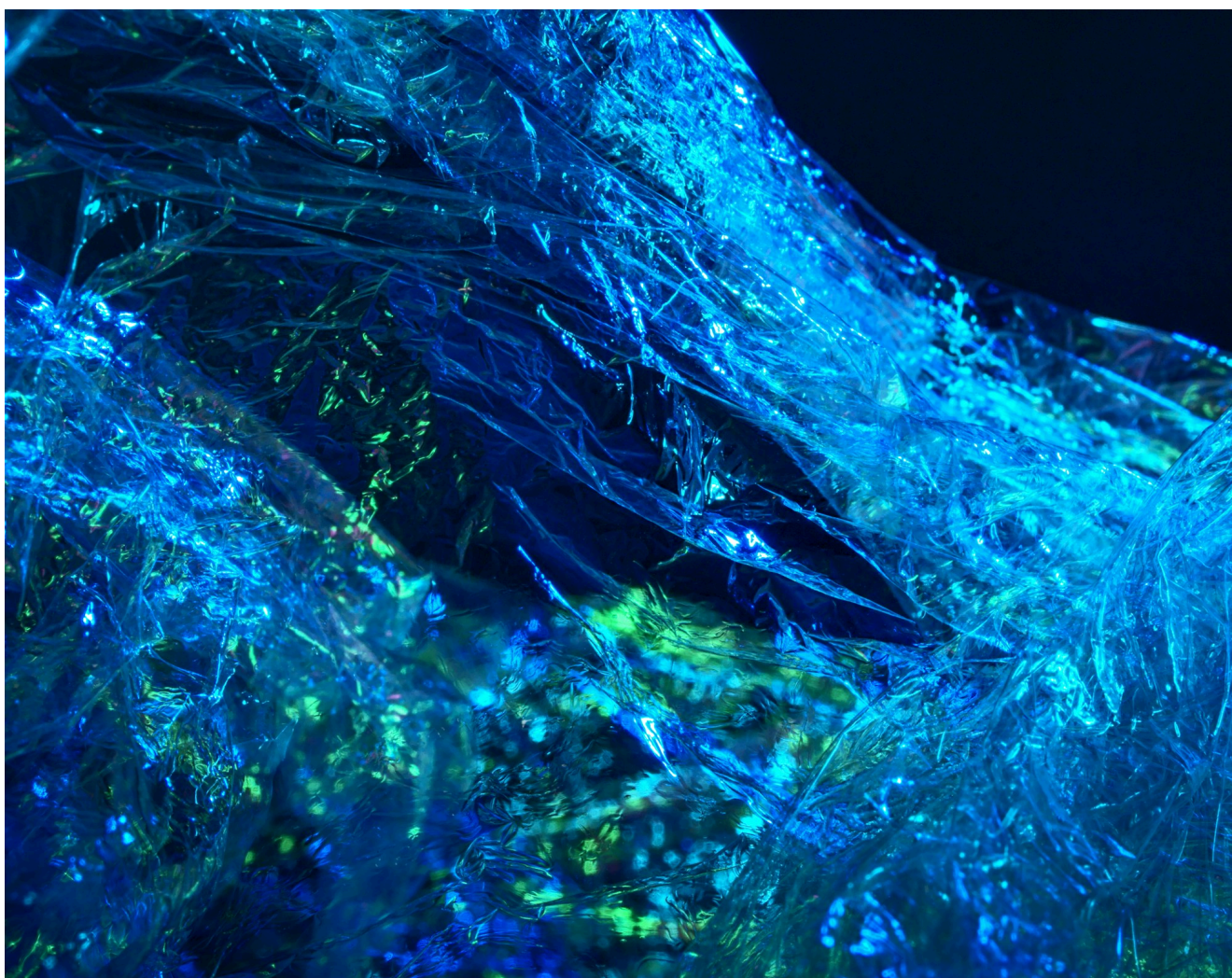


Photo by [Michael Dziedzic](#) / [Unsplash](#)

Introduction

Data Collection Rules theoretically

Subscribe

What are DCRs

DCR use cases regarding Microsoft Sentinel

Structures of Data Collection Rules

Data ingestion flow in Microsoft Sentinel

Custom data ingestion and transformations

How transformations work

Using multiple destinations

Costs for transformations

Creating transformations

Creating transformations for AMA sources

Creating transformations for standard tables

Why DCRs are so complicated

Logstash and DCRs

Golden tip

Introduction

Data Collection Rules in Azure can be very confusing once you really start using all of their potential. During my experience using them in complex setups, I identified a couple of pitfalls I would like to tackle for you. These pitfalls include the complex structures and types of DCRs, which type and structure of DCR you need for which use case, what the limitations and possibilities are of the different DCRs, and how you need to create the different DCRs.

In this post, you will learn what DCRs are, what structures and flavors there are, and how you can overcome the pitfalls mentioned earlier. All of this will be focussed on creating data collection rules for Microsoft Sentinel use cases.

Data Collection Rules theoretically

To clarify how data collection rules work, we will first have to go over some basics.

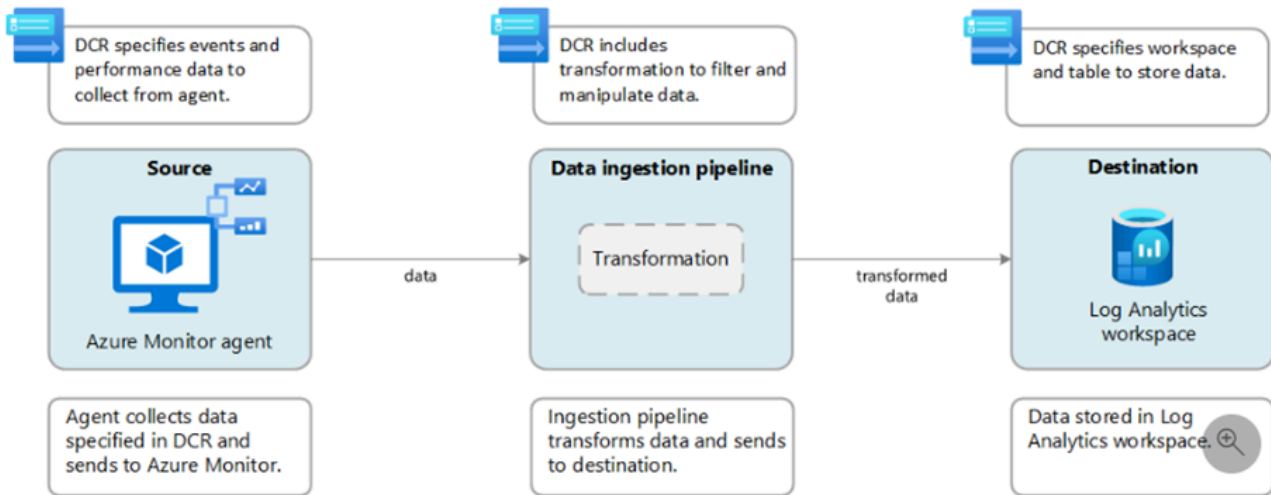
What are DCRs

Data Collection Rules are Azure resources that define the data collection process in Azure Monitor. It defines the details of a particular data collection scenario including which data should be collected, how to transform the data, and where to send that data. It is the new way of ingesting data into multiple types of destinations in Azure. This reflects in the fact that for example all Microsoft Sentinel data connectors that use the MMA agent or Log Analytics workspace ID and Key, are legacy in the Azure portal. When you try to create a data connector using the new way, you will find yourself creating Data Collection Rule in most cases.

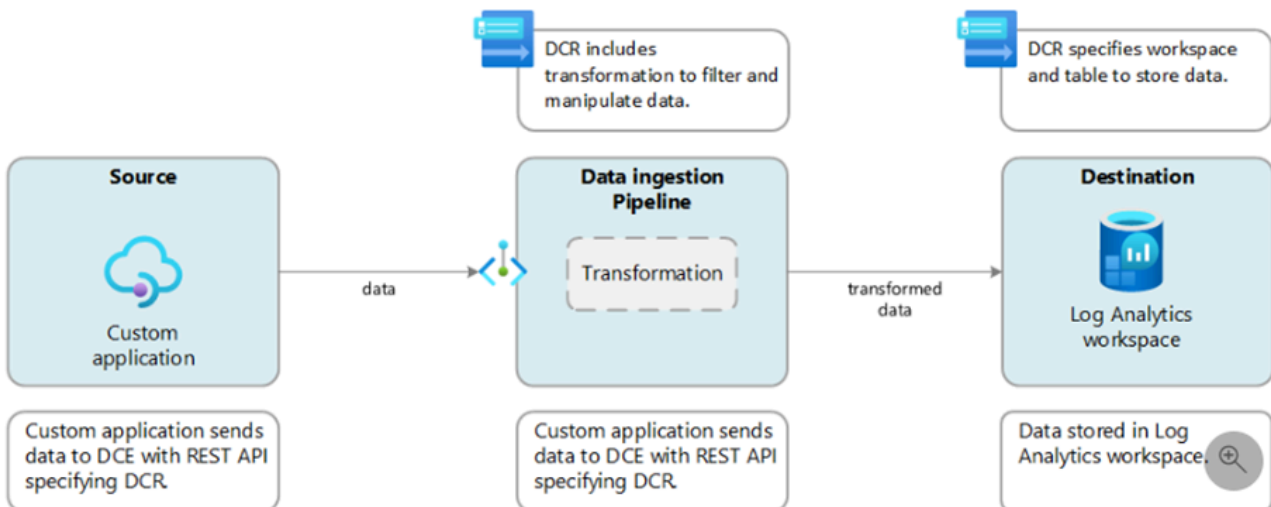
DCR use cases regarding Microsoft Sentinel

For Microsoft Sentinel, there are a couple of primary use cases for which you can use Data Collection Rules:

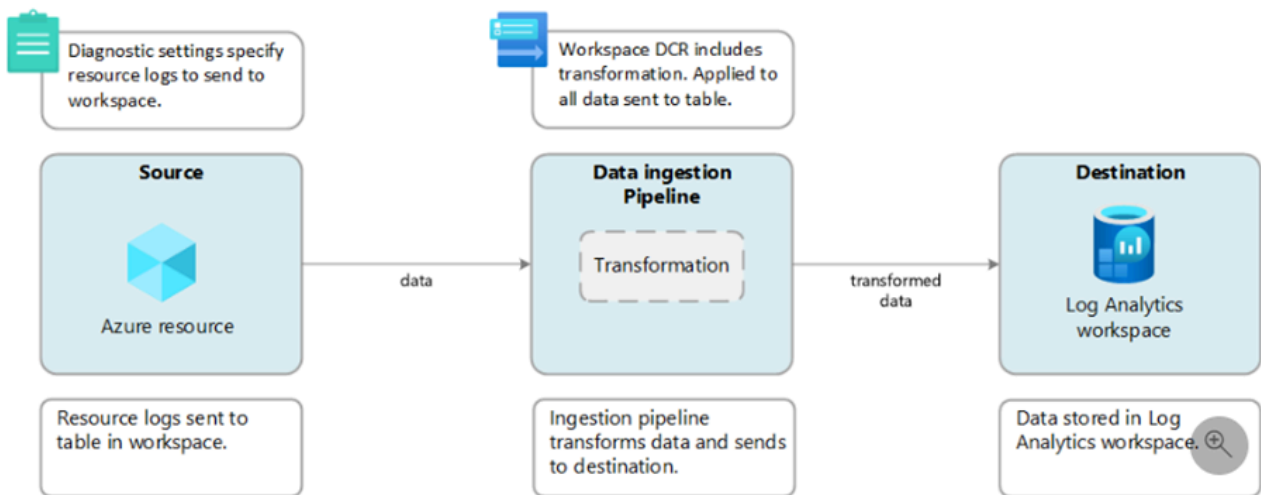
- Azure Monitor Agent – You can create DCR rules with an association to an Azure Monitor Agent, to receive the data from an AMA agent and send it to a Log Analytics Workspace. This can be done for Windows Events, Linux Syslog events, or third-party syslog forwarding via a Syslog server



- Custom Logs – You can send data to the REST API that is connected to a Data Collection Endpoint which is linked to a Data Collection Rule.



- Workspace transformations – When you want to use transformations for a legacy workload that does not support DCR or Azure service connectors, you can associate a DCR to the workspace and use it for supported tables.



Regardless of the source that is used for the DCRs, they are **very powerful for filtering and normalization** of data. Since DCRs are available, you have a lot of power to choose

- Which data can be forwarded to Sentinel
- How data must be normalized to fit in a certain table
- Under which complex conditions an event must be dropped (using KQL conditions instead of filtering at the source, which is generally very limited)

Structures of Data Collection Rules

There are two main structures for creating data collection rules, which are the **'Custom Logs' structure** and the **'Azure Monitor Agent' structure**. I found in practice that there is no hard limit between these structures, which means you can combine these structures in one ARM template (not recommended though). After you read the below, keep in mind that in essence, the big difference between the two structures is that **Custom Logs uses streamDeclarations and need a Data Collection Endpoint** and that **Azure Monitoring Agent uses DataSources**

instead of streamDeclarations and does not need a Data Collection Endpoint.

Custom logs

A Custom Logs DCR contains the following sections:

- StreamDeclarations – This section contains the different types of data that will be sent via the HTTP endpoint. Each stream is an object with a Key and a value. The Key is the stream name, which needs to start with 'Custom-'. The value is a list of top-level properties that are contained in the JSON data that will be sent.

```
"streamDeclarations": {  
  "Custom-testtable_CL": {  
    "columns": [  
      {  
        "name": "TimeGenerated",  
        "type": "datetime"  
      },  
      {  
        "name": "Key1",  
        "type": "string"  
      },  
      {  
        "name": "Key2",  
        "type": "int"  
      },  
      {  
        "name": "Key3",  
        "type": "boolean"  
      },  
      {  
        "name": "Key4",  
        "type": "string"  
      }  
    ]  
  }  
},
```

- Destinations – The destinations to where the data needs to be sent (yes, multiple are possible)

```

"destinations": {
  "logAnalytics": [
    {
      "workspaceResourceId": "[parameters('workspaces_rvd_weu_sentinelbase_la_externalid')]",
      "name": "042c76ff99d44b20838e866acc1c8e1f"
    }
  ]
},

```

- DataFlows – This section ties the other sections together. It defines the following properties for each stream declared in streamDeclarations
- Streams – Takes the streamsDeclarations
- Destination – Takes the destinations
- TransformKql – kql used to transform data from the streamDeclarations to the destinations
- OutputStream – which table in the workspace the data will be sent to. The value needs to be 'Microsoft-[tableName]' when data is being ingested in standard Log Analytics tables, or 'Custom-[tableName]' when data is ingested in a custom-created table

```

"dataFlows": [
  {
    "streams": [
      "Custom-testtable_CL"
    ],
    "destinations": [
      "042c76ff99d44b20838e866acc1c8e1f"
    ],
    "transformKql": "source | extend TimeGenerated = now()",
    "outputStream": "Custom-testtable_CL"
  }
]

```

Azure Monitor Agent

A DCR for Azure Monitor Agent contains the following sections

- DataSources – Contains the unique source of monitoring data in its own format. Each data source has its own data source type, where each type defines its own unique set of properties that must be specified. The currently available data source types are:
- Extension – VM extensions used by Log Analytics solutions and Azure services
- PerformanceCounters – For both Windows and Linux
- Syslog – Syslog events on Linux
- WindowsEventLogs – Windows Events

More info about the data source types and their properties can be found in the sample ARM template at <https://learn.microsoft.com/en-us/azure/azure-monitor/agents/data-collection-rule-sample-agent>

```
"dataSources": {  
  "windowsEventLogs": [  
    {  
      "streams": [  
        "Microsoft-SecurityEvent"  
      ],  
      "xpathQueries": [  
        "Security!*",  
        "Microsoft-Windows-AppLocker/EXE and DLL!*",  
        "Microsoft-Windows-AppLocker/MSI and Script!*"br/>      ],  
      "name": "eventLogsDataSource"  
    }  
  ]  
},
```

- Destination – Destinations where data should be sent to


```

"destinations": {
  "logAnalytics": [
    {
      "workspaceResourceId": "[parameters('workspaces_rvd_weu_sentinelbase_la_externalid')]",
      "name": "DataCollectionEvent"
    }
  ]
},

```

- DataFlows – Indicate which streams should be sent to which destinations (can have all of the properties like the DataFlows part described in the Custom Logs format)

```

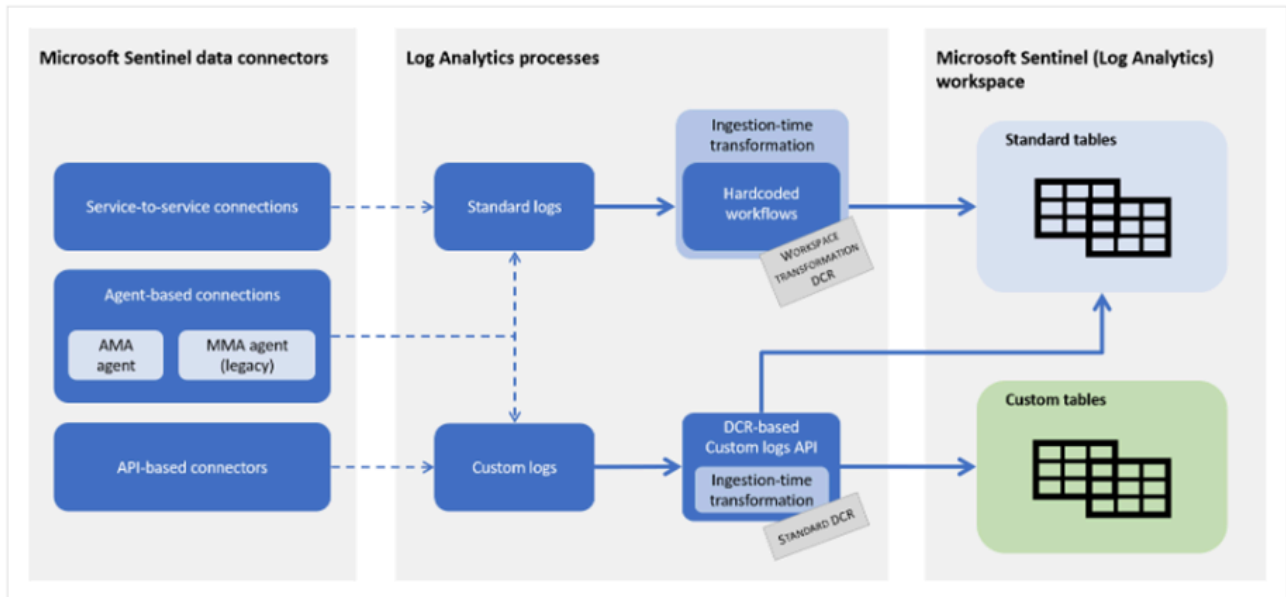
"dataFlows": [
  {
    "streams": [
      "Microsoft-SecurityEvent"
    ],
    "destinations": [
      "DataCollectionEvent"
    ]
  }
]

```

- Streams - This unique handle describes a set of data sources that will be transformed and schematized as one type. Each data source requires one or more streams, and one stream can be used by multiple data sources. All data sources in a stream share a common schema. Use multiple streams, for example, when you want to send a particular data source to multiple tables in the same Log Analytics workspace. This is defined in both the DataSources and the DataFlows

Data ingestion flow in Microsoft Sentinel

Data ingestion flow in Microsoft Sentinel can be a little overwhelming when you are first learning about it. In the [Microsoft documentation](#) you will find a schema that can help you with understanding the different ingestion flows:



Below, we will go over the different flows based on the data connectors that are found at the left of the schema.

Service-to-service connections

When you would like to create transformations for service-to-service connections, your only option is to create a **workspace transformation DCR**. More info about how to create workspace transformation DCRs and what they are can be found later in the blog post.

Agent-based connections

Agent-based connections can use **both a workspace transformation DCR and a standard DCR**. When you use a workspace transformation, you are creating a DCR formed in the 'Azure Monitor Agent' format, without a DataSource configured in the DCR. By doing this, the DCR

transforms data for all data being ingestion in a specific table, without caring what the data source is (as long as it is not another DCR).

When you create a standard DCR, you can create transformation for specific data sources. The structure that is being used here is also the 'Azure Monitor Agent' structure of the DCR.

API-based connectors

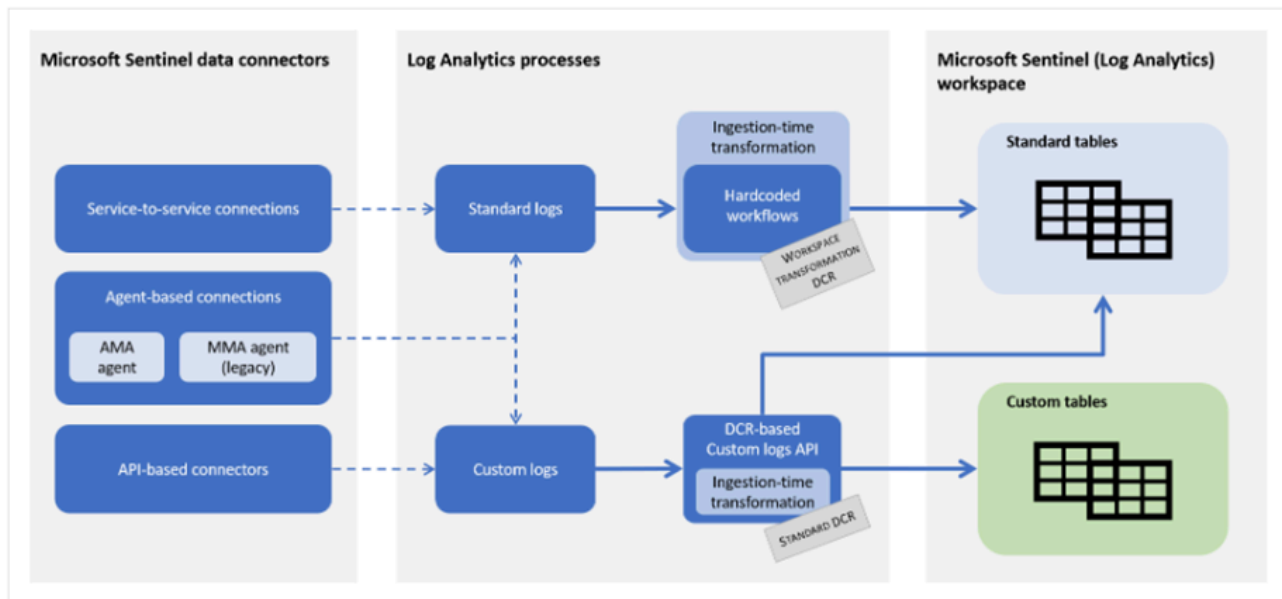
API-based connectors use **Standard DCRs** to send and transform data in tables. These DCRs are always created in the 'Custom logs' DCR format, which also means that there is a **Data Collection Endpoint needed for these DCRs**.

Custom data ingestion and transformations

Going forward, we will be focusing a little bit more on the custom data ingestion part of the Data Collection Rules.

How transformations work

We already mentioned that you can create transformation in DCRs to manipulate your data and that this must be done either by workspace transformation DCRs or by standard DCRs. Below you will find more info about both of them. Keep in mind the below schema while learning about them:



Workspace transformation DCR

When you create a workspace transformation DCR, you will find that the DCR is created in the structure of – what Microsoft calls – an **Azure Monitor Agent structure**. This DCR will have the DataSources, Destinations, DataFlows, and Stream properties, but the DataSources field will remain empty.

It is important to note that a workspace can only have one workspace transformation DCR, but that DCR can contain separate transformations for each input stream. Below you will find a workspace transformation DCR created for the AADNonInteractiveUserSignInLogs table and the SecurityEvent table:

```

"kind": "WorkspaceTransforms",
"properties": {
  "dataSources": {},
  "destinations": {
    "logAnalytics": [
      {
        "workspaceResourceId": "[parameters('workspaces_rvd_weu_sentinelbase_la_externalid')]",
        "name": "042c76ff99d44b20838e866acc1c8e1f"
      }
    ]
  },
  "dataFlows": [
    {
      "streams": [
        "Microsoft-Table-AADNonInteractiveUserSignInLogs"
      ],
      "destinations": [
        "042c76ff99d44b20838e866acc1c8e1f"
      ],
      "transformKql": "source"
    },
    {
      "streams": [
        "Microsoft-Table-SecurityEvent"
      ],
      "destinations": [
        "042c76ff99d44b20838e866acc1c8e1f"
      ],
      "transformKql": "source"
    }
  ]
}

```

This type of DCR does not care what the source of the data is, as long as the data is not coming from another DCR. The supported tables that can be used for workspace transformation DCRs are currently limited and can be found [here](#). **Workspace transformation DCR can only be used for transformations in standard tables.**

Standard DCR

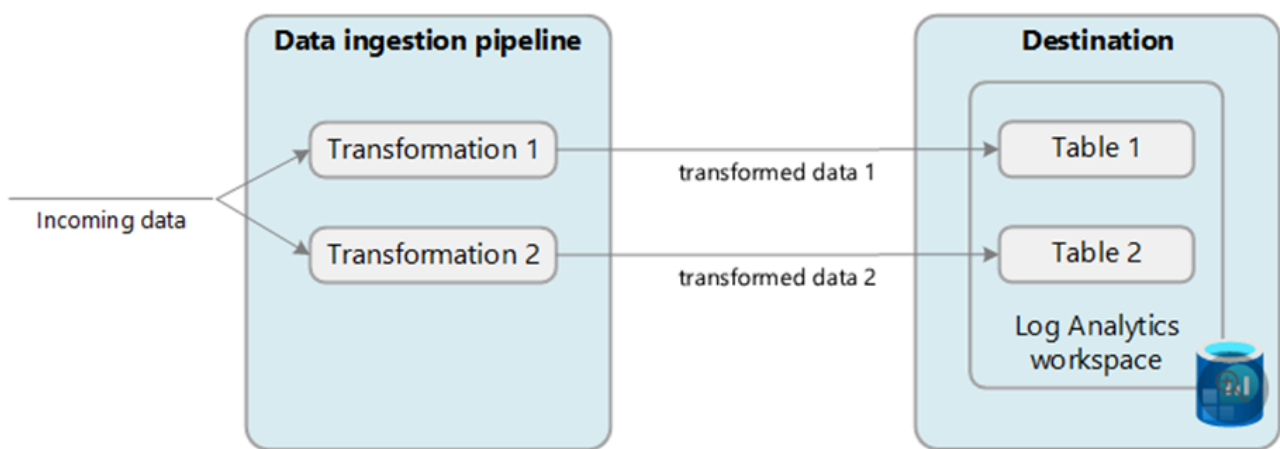
A standard DCR can be created in **both the Custom Logs and Azure Monitor Agent structure**. The important part here is that each connector or log source can have its own dedicated DCR, though multiple connectors or sources can share a common standard DCR as well. These DCRs are currently only supported for AMA agents and workflows using the Log Ingestion API but can **send data to both custom tables and a couple of standard tables**. The supported standard tables are currently

limited and can be found [here](#), at the time of writing the supported standard tables are:

- CommonSecurityLog
- SecurityEvents
- Syslog
- WindowsEvents

Using multiple destinations

With transformation, you can send data to multiple destinations in a Log Analytics Workspace, where you can provide a KQL transformation for each destination. Currently, the tables in the DCR must reside in the same Log Analytics Workspace.



An example of a use case for sending data to multiple destinations is when you want to send debug and verbose logs to a custom table configured as basic logs, and other more import severities to an analytics table for interactive querying.

Costs for transformations

While transformations themselves don't incur direct costs, some scenarios can result in additional charges. These are described and explained in the Microsoft documentation: <https://learn.microsoft.com/en-us/azure/azure-monitor/essentials/data-collection-transformations#cost-for-transformations>

Creating transformations

You can create transformations and DCRs either by using the Azure portal or by deploying ARM templates via REST API or your favorite client libraries. In this section, we will be focussing on creating DCRs via the Azure portal, since I found this the easiest way to create a DCR (due to the complexity of the ARM structure of a DCR). Once we created them via the portal, we can then export the ARM templates for later use in DevOps scenarios.

Creating a workspace transformation DCR

If you still remember, there can only be one workspace transformation DCR. This DCR can be used for transformations from service-to-service connectors or the AMA agent, to standard and supported Log Analytics tables. To create a workspace transformation, go to your **Log Analytics workspace > Tables**. Here you will find all the tables present in your workspace.

rvd-weu-sentinelbase-1a | Tables

Log Analytics workspace

Search

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Logs
Settings

Tables

Agents
Usage and estimated costs
Data export
Network isolation
Linked storage accounts
Properties
Locks
Classic
Legacy agents management
Legacy activity log connector
Legacy storage account logs
Legacy computer groups
Legacy solutions
System center
Workspace summary (deprecated)

For the list of tables supporting ingestion-time transformations please refer to [documentation](#)

Create Delete

Filter by name Type: All Plan: All

Showing 84 results No grouping

Table name	Type	Plan	Interactive retention	Archive period
AADManagedIdentitySignInLogs	Azure table	Analytics	Workspace default (30 days)	...
AADNonInteractiveUserSignInLogs	Azure table	Analytics	Workspace default (30 days)	...
AADServicePrincipalSignInLogs	Azure table	Analytics	Workspace default (30 days)	...
Alert	Azure table	Analytics	Workspace default (30 days)	...
AlertEvidence	Azure table	Analytics	Workspace default (30 days)	...
AlertInfo	Azure table	Analytics	Workspace default (30 days)	...
Anomalies	Azure table	Analytics	Workspace default (30 days)	...
AppCenterError	Azure table	Analytics	Workspace default (30 days)	...
ASimDnsActivityLogs	Azure table	Analytics	Workspace default (30 days)	...
ASimNetworkSessionLogs	Azure table	Analytics	Workspace default (30 days)	...
ASimWebSessionLogs	Azure table	Analytics	Workspace default (30 days)	...
AuditLogs	Azure table	Analytics	Workspace default (30 days)	...
AWSCloudTrail	Azure table	Analytics	Workspace default (30 days)	...
AWSCloudWatch	Azure table	Analytics	Workspace default (30 days)	...
AWSGuardDuty	Azure table	Analytics	Workspace default (30 days)	...

When you click on the three dots on the right side of the table, you will be able to create a transformation

Table name	Type	Plan	Interactive retention	Archive period
AADManagedIdentitySignInLogs	Azure table	Analytics	Workspace default (30 days)	...
AADNonInteractiveUserSignInLogs	Azure table	Analytics	Workspace default (30 days)	...
AADServicePrincipalSignInLogs	Azure table	Analytics	Workspace default (30 days)	...
Alert	Azure table	Analytics	Workspace default (30 days)	...
AlertEvidence	Azure table	Analytics	Workspace default (30 days)	...

Manage table

Create transformation

Edit schema

On the first page, you will be able to create a DCR. Note that if you already have a workspace transformation DCR, you will have to use that one (since only one workspace transformation DCR can exist)

AADNonInteractiveUserSignInLogs transformation ...

- 1 Basics 2 Schema and transformation 3 Review

Table details

Table name	AADNonInteractiveUserSignInLogs
Description	Non-interactive Azure Active Directory sign-in logs from user.

Data collection rule

Data collection rules (DCR) define the data coming into Azure Monitor and specify where that data should be sent or stored. [Learn more](#)

Data collection rule workspace-dcr-test

In the schema and transformation tab, you will be presented with the data that is currently present in the table. This is very handy to create your transformation query since you can interactively test the query with the current data in your table. Once you have developed the query, you can save the transformation and create the DCR.

AADNonInteractiveUserSignInLogs transformation ...

1 Basics 2 Schema and transformation 3 Review

Transformation editor

A transformation has already been done on this table. You can continue editing this transformation or delete it and start a new one.

TenantId	SourceSystem	TimeGenerated	OperationName	OperationVersion	Category
042c76ff...	Azure AD	2023-05-16T05:57:15...	Sign-in activity	1.0	NonInteractive...
042c76ff...	Azure AD	2023-05-16T08:50:54...	Sign-in activity	1.0	NonInteractive...
042c76ff...	Azure AD	2023-05-16T08:51:14...	Sign-in activity	1.0	NonInteractive...
042c76ff...	Azure AD	2023-05-16T08:52:51...	Sign-in activity	1.0	NonInteractive...
042c76ff...	Azure AD	2023-05-16T09:44:12...	Sign-in activity	1.0	NonInteractive...
042c76ff...	Azure AD	2023-05-16T09:52:31...	Sign-in activity	1.0	NonInteractive...
042c76ff...	Azure AD	2023-05-14T00:19:41...	Sign-in activity	1.0	NonInteractive...
042c76ff...	Azure AD	2023-05-14T10:00:23...	Sign-in activity	1.0	NonInteractive...
042c76ff...	Azure AD	2023-05-14T10:02:51...	Sign-in activity	1.0	NonInteractive...
042c76ff...	Azure AD	2023-05-14T10:12:56...	Sign-in activity	1.0	NonInteractive...

Transformation editor

1 source
2 extend DemoColumn = "Demo"

TimeGenerated [UTC]	DemoColumn	AppDisplayName	AppId	AuthenticationContextClass...	Authenticati...
5/16/2023, 5:57:15 AM	Demo	Windows Defender ATP for Flow	7ba73825-1ea2-4c07-a38c-90c...		
5/16/2023, 8:50:54 AM	Demo	Azure Portal	c44b4083-3600-49c1-b479-97...		
5/16/2023, 8:51:14 AM	Demo	Azure Portal	c44b4083-3600-49c1-b479-97...		
5/16/2023, 8:52:51 AM	Demo	Azure Portal	c44b4083-3600-49c1-b479-97...		
5/16/2023, 9:44:12 AM	Demo	App Service	7ba73825-1ea2-4c07-a38c-90c...		
5/16/2023, 9:52:31 AM	Demo	Azure Portal	c44b4083-3600-49c1-b479-97...		
5/14/2023, 9:19:41 AM	Demo	Azure Portal	c44b4083-3600-49c1-b479-97...		
5/14/2023, 10:00:23 AM	Demo	Microsoft_OperationsManage...	6e09b31f-0604-4c93-b914-e08...		
5/14/2023, 10:02:51 AM	Demo	Microsoft_OperationsManage...	6e09b31f-0604-4c93-b914-e08...		
5/14/2023, 10:12:56 AM	Demo	ADfsLocal	74d58136-14ec-4630-a99e-25...		

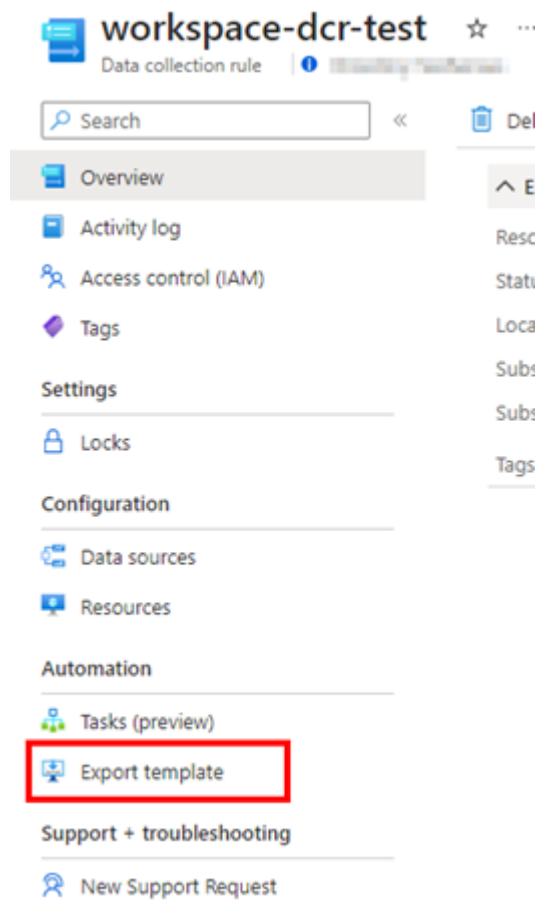
0.318ms | Display time (UTC+0000) | 1 - 10 of 10

Apply Cancel

Continue in Log Analytics

When you go to the Data Collection Rule in the Azure Portal, you can view and export the ARM template. Here you will find your DCR with the

transformations for every table:




```

"dataFlows": [
  {
    "streams": [
      "Microsoft-Table-SecurityEvent"
    ],
    "destinations": [
      "042c76ff99d44b20838e866acc1c8e1f"
    ],
    "transformKql": "source"
  },
  {
    "streams": [
      "Microsoft-Table-AuditLogs"
    ],
    "destinations": [
      "042c76ff99d44b20838e866acc1c8e1f"
    ],
    "transformKql": "source\n| extend Id = \"1\"\n"
  },
  {
    "streams": [
      "Microsoft-Table-AADNonInteractiveUserSignInLogs"
    ],
    "destinations": [
      "042c76ff99d44b20838e866acc1c8e1f"
    ],
    "transformKql": "source\n| extend DemoColumn_CF = \"Demo\"\n"
  }
]

```

Creating a standard DCR for AMA option 1

There are two experiences for creating standard DCRs for AMA sources (regarding Sentinel). The first one is by creating the DCR on the DCR page via the create button.

Data collection rules 🔍 ✕

[+ Create](#) [Manage view](#) [Refresh](#) [Export to CSV](#) [Open query](#) [Assign tags](#) [Delete](#)

Filter for any field... [Subscription equals all](#) [Resource group equals all](#) [Location equals all](#) [Add filter](#)

Showing 1 to 3 of 3 records. No grouping List view

Name	Subscription	Resource group	Location	Data sources	Destinations
<input type="checkbox"/> workspace-dcr-test	RVDD	RVDD-WEU-SENTINELBASE-rg	West Europe		Azure Monitor Logs
<input type="checkbox"/> Deploy-AMA	RVDD	RVDD-Research-rg	West Europe	Performance counters, Windows event logs	Azure Monitor Logs, Azure Monitor Metrics
<input type="checkbox"/> dc-arc-prod-westeurope-001	RVDD	RVDD-WEU-SENTINELBASE-rg	West Europe	Windows event logs	Azure Monitor Logs

Here you start configuring the name, region, and resource group of the DCR. Depending on the OS of the device, you choose Linux or Windows. A Data Collection Endpoint is not needed for Windows Logs, Syslog logs, or Performance counters:

Basics Resources Collect and deliver Review + create

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all of your resources.

Rule details

Rule Name * ✓

Subscription * ⓘ ▼

Resource Group * ⓘ ▼
[Create new](#)

Region * ⓘ ▼

Platform Type * ⓘ ☒ Windows ☐ Linux

A Data Collection Endpoint is optional for collecting Windows Event Logs, Linux Syslog or Performance Counters. It is required for all other data sources.

Data Collection Endpoint ⓘ ▼

On the next page, you choose the resource from where you want to collect logs

Basics **Resources** Collect and deliver Review + create

Pick a set of machines to collect data from. The Azure Monitor Agent will be automatically installed on virtual machines, scale sets, and Arc-enabled servers. download the client installer and follow the [guidance](#)

i This will also enable System Assigned Managed Identity on these machines, in addition to existing User Assigned Identities (if any).

[+ Add resources](#) [+ Create endpoint](#)

Enable Data Collection Endpoints ⓘ ☐

i Only virtual machines in the same region can be assigned to the same endpoint. [Learn more about event logs and XPath syntax](#) ⓘ

Name	Type	Location	Resource group	Subscription
WIN-PELTVJHQN52	Server - Azure Arc	West Europe	rg-arc-prod-westeurope-001	RVDD

In the Collect and Deliver tab, you can configure the types of logs you want to collect and where the logs need to be sent. Depending on whether you choose Linux or Windows as OS, this page will look different.

Create Data Collection Rule ...

Data collection rule management

Basics Resources **Collect and deliver** Review + create

Configure which data sources to collect, and where to send the data to.

+ Add data source

Data source Destination(s)

No standard data sources or destinations found.

✖ This data collection rule doesn't have any data sources or destinations selected.

Review + create < Previous Next : Review + create >

Data source Destination

Select which data source type and the data to collect for your resource(s).

Data source type *

Windows Event Logs

! If using Sentinel, use the [connector configuration](#) for collecting Windows Security events to avoid unexpected increase in storage cost. [Learn more](#)

Choose Basic to enable collection of event logs. Choose Custom if you want more control over which event logs are collected.

None **Basic** Custom

Configure the event logs and levels to collect:

Application

☐ Critical

☐ Error

☐ Warning

☐ Information

☐ Verbose

Security

☒ Audit success

☒ Audit failure

System

☐ Critical

☐ Error

☐ Warning

☐ Information

☐ Verbose

Add data source Next : Destination > Cancel

When you finish these configurations, you can create the DCR and check the ARM template in the Azure portal.

standard-dcr-ama-test | Export template ...

Data collection rule

Download Add to library Deploy Visualize template

To export related resources, select the resources from the Resource Group view then select the "Export template" option from the tool bar.

☒ Include parameters

Template Parameters Scripts

```

21  {
22    "kind": "Windows",
23    "properties": {
24      "dataSources": {
25        "windowsEventLogs": [
26          {
27            "streams": [
28              "Microsoft-Event"
29            ],
30            "xpathQueries": [
31              "Security!*[System[(band(Keywords,1351879888211488))]]"
32            ],
33            "name": "eventLogsDataSource"
34          }
35        ],
36        "destinations": {
37          "logAnalytics": [
38            {
39              "workspaceResourceId": "[parameters('workspaces_rvd_wau_sentinelbase_la_externalid')]",
40              "name": "la--1641389435"
41            }
42          ]
43        },
44        "dataFlows": [
45          {
46            "streams": [
47              "Microsoft-Event"
48            ],
49            "destinations": [
50              "la--1641389435"
51            ]
52          }
53        ]
54      }
55    }
56  }







```

You might be wondering why we did not have the option to create a transformation in the portal while we were creating the DCR. And in fact, I do not know either. I suspect that this has something to do with the fact that the logs of an Azure Monitoring Agent are being sent to standard tables by default. However, this does not mean that transformations are

not supported for standard DCRs with AMA sources. Later in the post, we will be going over how we can create transformation anyway.

Creating a standard DCR for AMA option 2

The second option for creating AMA-related DCRs is by going to the Microsoft Sentinel Data Connectors blade. When we search for AMA, we get three options:

Status	Connector name ↑
	Akamai Security Events (Preview) Akamai
	Amazon Web Services Amazon
	Amazon Web Services S3 (Preview) Amazon
	Common Event Format (CEF) via AMA (Preview) Microsoft
	Windows DNS Events via AMA (Preview) Microsoft
	Windows Security Events via AMA Microsoft

Depending on what type of data you want to ingest, you select one of the three data connectors. The differences between the DCRs created by the different data connectors are very little since they are all **Standard DCRs in the Azure Monitoring Agent format**. What is different is the user experience in the portal, since you will have other configuration options for Windows Logs than you have for CEF forwarding for example.

Windows Security Logs:

When you open the Windows Security Events via the AMA page, you will be able to create a DCR:

The screenshot displays the 'Windows Security Events via AMA' page in Microsoft Sentinel. The left sidebar contains a description of the connection, a status bar showing 'Connected' and 'Microsoft Provider', and a chart titled 'Data received' showing a line graph of data received over time (May 15 to May 19). The right pane shows the 'Instructions' tab with sections for 'Prerequisites' and 'Configuration'. The 'Prerequisites' section lists requirements for integrating with Windows Security Events via AMA. The 'Configuration' section includes a 'Refresh' button and a table of existing data collection rules.

Rule name	Created by	Event filter type
dc-arc-prod-west-europe-001	Sentinel	AllEvents

+ Create data collection rule

Here you will be able to create a DCR for Windows Servers just like you can in option 1, but you will find that you have other options in the 'Collect' tab to configure which events you want to ingest:

Basics Resources Collect Review + create

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all of your resources.

Rule details

Rule Name *	dc-arc-tst-west-europe-001 ✓
Subscription * ⓘ	RVDD ✓
Resource Group * ⓘ	RVD-WEU-SENTINELBASE-rg ✓

Basics Resources Collect Review + create

Select which events to stream. ⓘ

☒ All Security Events ☐ Common ☐ Minimal ☐ Custom

The reason behind this is that we are configuring a DCR for a specific use case, which is ingesting Security related events rather than all events from a specific severity or class. This does not mean the DCR will be in other formats, the portal just tries to help you create a DCR for your use case.

Common Event Format via AMA:

Another example is the CEF forwarding via AMA. In this data connector, you will also be able to create your DCR.

Common Event Format (CEF) via AMA (Preview)

Not connected

Microsoft

--

Status

Provider

Last Log Received

Description

Common Event Format (CEF) is an industry standard format on top of Syslog messages, used by many security vendors to allow event interoperability among different platforms. By connecting your CEF logs to Microsoft Sentinel, you can take advantage of search & correlation, alerting, and threat intelligence enrichment for each log.

Last data received

--

Related content

0

Workbooks

1

Queries

0

Analytics rules templates

Data received [Go to log analytics](#)

Total data received

0

Data types

CommonSecurityLog --

Instructions

Prerequisites

To integrate with Common Event Format (CEF) via AMA (Preview) make sure you have:

- ✓ **Workspace data sources:** read and write permissions.
- ℹ To collect data from non-Azure VMs, they must have Azure Arc installed and enabled. [Learn more](#)

Configuration

Enable data collection rule

CEF Events logs are collected only from **Linux** agents.

[Refresh](#) ⓘ

Rule name **Event filter type**

No data collection rule found

[+Create data collection rule](#)

Run the following command to install and apply the CEF collector:

```
sudo wget -O Forwarder_AMA_installer.py https://raw.githubusercontent.com/...
```

Here you will be able to create your DCR for CEF forwarding, where you will find that the Collect tab has other options again in comparison to creating Windows Security Events DCRs:

Basics Resources **Collect** Review + create

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all of your resources.

Rule details

Rule Name *	dc-arc-tst-weteurope-001	✓
Subscription * ⓘ	RVDD	▼
Resource Group * ⓘ	RVD-WEU-SENTINELBASE-rg	▼

Basics Resources **Collect** Review + create

Select which data source type and the data to collect for your resource(s).

Facility	Minimum log level
LOG_AUTH	none ▼
LOG_AUTHPRIV	none ▼
LOG_CRON	none ▼
LOG_DAEMON	none ▼
LOG_MARK	none ▼
LOG_KERN	none ▼
LOG_LOCAL0	none ▼
LOG_LOCAL1	none ▼
LOG_LOCAL2	none ▼
LOG_LOCAL3	none ▼
LOG_LOCAL4	none ▼
LOG_LOCAL5	none ▼
LOG_LOCAL6	none ▼
LOG_LOCAL7	LOG_ERR ▼
LOG_LPR	none ▼
LOG_MAIL	none ▼
LOG_NEWS	none ▼
LOG_SYSLOG	none ▼
LOG_USER	none ▼

Again, this does not mean the DCR will be in other formats, the portal just tries to help you create a DCR for your use case.

Comparing the Windows and CEF DCRs

Now that we created DCRs for the different data connectors, we will compare the templates that were created by them.

When we check the DCR for **Windows Security Events**, we see that the data connector created a stream to ingest the data in the SecurityEvents table. This makes it easy for us since we do not have to search ourselves which table is best suiting for Windows Security Logs. We also see that the data connector created an XPath query based on the level of event we choose to ingest in the 'Collect' tab, which makes sure we do not have to decide which events we want to ingest:



```

"properties": {
  "dataSources": {
    "windowsEventLogs": [
      {
        "streams": [
          {
            "name": "Microsoft-SecurityEvent"
          }
        ],
        "xpathQueries": [
          "Security![System[(EventID=1102) or (EventID=4624) or (EventID=4625) or (EventID=4657) or (EventID=4663) or (EventID=4688) or (EventID=4700) or (EventID=4702) or (EventID=4719) or (EventID=4720) or (EventID=4722) or (EventID=4723) or (EventID=4724) or (EventID=4727) or (EventID=4728)]]]",
          "Security![System[(EventID=4732) or (EventID=4735) or (EventID=4737) or (EventID=4739) or (EventID=4740) or (EventID=4754) or (EventID=4755) or (EventID=4756) or (EventID=4767) or (EventID=4799) or (EventID=4825) or (EventID=4946) or (EventID=4948) or (EventID=4956) or (EventID=5024)]]]",
          "Security![System[(EventID=5033) or (EventID=8222)]]",
          "Microsoft-Windows-AppLocker/EXE and DLL[System[(EventID=8001) or (EventID=8002) or (EventID=8003) or (EventID=8004)]]",
          "Microsoft-Windows-AppLocker/MSI and Script[System[(EventID=8005) or (EventID=8006) or (EventID=8007)]]"
        ],
        "name": "eventLogsDataSource"
      }
    ]
  },
  "destinations": {
    "logAnalytics": [
      {
        "workspaceResourceId": "[parameters('workspaces_rvd_uwu_sentinelbase_la_externalId')]",
        "name": "DataCollectionEvent"
      }
    ]
  },
  "dataFlows": [
    {
      "streams": [
        {
          "name": "Microsoft-SecurityEvent"
        }
      ],
      "destinations": [
        "DataCollectionEvent"
      ]
    }
  ]
}

```

When we check the CEF DCR template, we see that the data connector now created a stream to send events to the CommonSecurityLog table, and changed the data sources to Syslog with the syslog facilities that we choose in the 'Collect' tab

```
"dataSources": {  
  "syslog": [  
    {  
      "streams": [  
        "Microsoft-CommonSecurityLog"  
      ],  
      "facilityNames": [  
        "local7"  
      ],  
      "logLevels": [  
        "Error",  
        "Critical",  
        "Alert",  
        "Emergency"  
      ],  
      "name": "sysLogsDataSource-1688419672"  
    }  
  ],  
  "destinations": {  
    "logAnalytics": [  
      {  
        "workspaceResourceId": "[parameters('workspaces_rvd_weu_sentinelbase_la_externalid')]",  
        "name": "DataCollectionEvent"  
      }  
    ]  
  },  
  "dataFlows": [  
    {  
      "streams": [  
        "Microsoft-CommonSecurityLog"  
      ],  
      "destinations": [  
        "DataCollectionEvent"  
      ]  
    }  
  ]  
}
```

When comparing bot templates, we can conclude that the **settings and configurations** of the DCRs are **different** and specifically created for a different use case, but that **both DCRs** are still **standard DCRs in the Azure Monitoring Agent format**.

Creating standard DCR for Log Ingestion API sources

When you want to ingest data via the Log Ingestion API, you need to create a DCR with a Data Collection Endpoint. To start creating such a DCR, you will need to go to the tables of your log analytics workspace, and create a custom log DCR:

rvd-weu-sentinelbase-la | Tables ☆ ...

Log Analytics workspace

Search

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Logs

Settings

Tables

Agents

Usage and estimated costs

Data export

For the list of tables supporting ingestion-time transformations please refer to [documentation](#)

+ Create ▾ | Delete

New custom log (DCR-based)

New custom log (MMA-based)

Type : All Plan : All

Showing 84 results

Table name ↑↓	Type ↑↓
<input type="checkbox"/> AADManagedIdentitySignInLogs	Azure table
<input type="checkbox"/> AADNonInteractiveUserSignInLogs	Azure table
<input type="checkbox"/> AADServicePrincipalSignInLogs	Azure table

In the Basic tab, you need to choose a table name where your events will flow to. If you pay attention, you will see that this is a custom table that will be created. This is perfect if you want to ingest data in a custom table, but as mentioned earlier you can also send data via a standard DCR to a couple of build-in tables. In this view, this is not something you can choose to do. However, there is a **workaround that we will cover later in this post**. For now, choose a table name, DCR name, and a Data Collection Endpoint that you will use:

1 Basics 2 Schema and transformation 3 Review

Table details

Start by adding a name and description for the table you're creating. On the next step, upload a sample of your custom log and adjust the table details to your needs.

Table name * ✓

Description

Data collection rule

Data collection rules (DCR) define the data coming into Azure Monitor and specify where that data should be sent or stored. [Learn more](#)

Data collection rule * [Create a new data collection rule](#)

Data collection endpoint *

In the Schema and Transformation tab, you will see that you need to upload a sample file of your data. This is because the portal needs to know the format of the data so it can be included in the template. If you are awake, you should know by now that this type of DCR will be a **standard DCR in the Custom Logs format**, since this format requires you to define your columns:

```

"streamDeclarations": {
  "Custom-testtable_CL": {
    "columns": [
      {
        "name": "TimeGenerated",
        "type": "datetime"
      },
      {
        "name": "Key1",
        "type": "string"
      },
      {
        "name": "Key2",
        "type": "int"
      },
      {
        "name": "Key3",
        "type": "boolean"
      },
      {
        "name": "Key4",
        "type": "string"
      }
    ]
  }
}

```

[Basics](#) [Schema and transformation](#) [Review](#)
[Upload sample file](#) [Transformation editor](#)

⚠ There was no timestamp field found in the sample provided. The transformation was updated to populate TimeGenerated column with the timestamp of data ingestion. Please use the transformation editor to review or update the logic of TimeGenerated column population in the destination table. [Learn more](#)

TimeGenerated	Key1	Key2	Key3	Key4
2023-05-20T07:42:35.739Z	String	2	true	key5=5 key6=6 key7=7

Once the file is uploaded, you can create the transformation in the GUI just like we did for the workspace transformation DCR:

Create a custom log

[Basics](#) [Schema and transformation](#) [Review](#)
[Upload sample file](#) [Transformation editor](#)

⚠ There was no timestamp field found in the sample provided. The transformation was updated to populate TimeGenerated column with the timestamp of data ingestion. Please use the transformation editor to review or update the logic of TimeGenerated column population in the destination table. [Learn more](#)

```

1 source
2 extend TimeGenerated = now()
3 parse Key4 with "key5=" Key5 "key6=" Key6 "key7=" Key7

```

TimeGenerated	Key1	Key2	Key3	Key4	Key5	Key6	Key7
2023-05-20T07:42:35.739Z	String	2	true	key5=5 key6=6 key7=7	5	6	7

If we check the DCR template again, we now see that the template is in the Custom Logs format, and includes the components we created via the portal:

```

"dataCollectionEndpointId": "[parameters('dataCollectionEndpoints_RSECTVS01_externalid')]",
"streamDeclarations": {
  "Custom-sampleData_CL": {
    "columns": [
      {
        "name": "TimeGenerated",
        "type": "datetime"
      },
      {
        "name": "Key1",
        "type": "string"
      },
      {
        "name": "Key2",
        "type": "int"
      },
      {
        "name": "Key3",
        "type": "boolean"
      },
      {
        "name": "Key4",
        "type": "string"
      }
    ]
  }
},
"dataSources": {},

```

```

"destinations": {
  "logAnalytics": [
    {
      "workspaceResourceId": "[parameters('workspaces_rvd_weu_sentinelbase_la_externalid')]",
      "name": "042c76ff99d44b20838e866acc1c8e1f"
    }
  ]
},
"dataFlows": [
  {
    "streams": [
      "Custom-sampleData_CL"
    ],
    "destinations": [
      "042c76ff99d44b20838e866acc1c8e1f"
    ],
    "transformKql": "source\n| extend TimeGenerated = now()\n| parse Key4 with * \"key5=\" Key5 \"|key6=\" Key6 \"|key7=\" Key7\n",
    "outputStream": "Custom-sampleData_CL"
  }
]
}

```

Creating transformations for AMA sources

If you remember the section where we created DCRs for the AMA connectors, we were not able to create a transformation KQL in the portal. This does not mean transformations are not supported for these DCRs. You can just add a transformation KQL and an outputStream to the ARM template:

```
    },  
    "dataFlows": [  
      {  
        "streams": [  
          "Microsoft-SecurityEvent"  
        ],  
        "destinations": [  
          "DataCollectionEvent"  
        ],  
        "transformKql": "source\n| extend TimeGenerated = now()",  
        "outputStream": "Microsoft-SecurityEvent"  
      }  
    ]  
  }
```

The only thing you need to make sure of is that the columns you use in the transformationKQL are in fact columns that are known for the DCR source table. When this is not the case, the deployment of the ARM template will fail. The table where the columns need to come from is the table that is defined in the streams part.

Creating transformations for standard tables

If you remember the section where we created a DCR for the Log Ingestion API, we mentioned that it was not possible in the GUI to choose a standard table, even though sending logs via a standard DCR support a couple of standard tables. To work around this issue, we can easily follow the below steps.

Create a DCR for Log Ingestion API as mentioned earlier:

rvd-weu-sentinelbase-la | Tables ☆ ...

Log Analytics workspace

Search

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Logs
- Settings
 - Tables**
 - Agents
 - Usage and estimated costs
 - Data export

For the list of tables supporting ingestion-time transformations please refer to [documentation](#)

+ Create ▾ | Delete

New custom log (DCR-based)
New custom log (MMA-based)

Type : All Plan : All

Showing 84 results

<input type="checkbox"/> Table name ↑↓	Type ↑↓
<input type="checkbox"/> AADManagedIdentitySignInLogs	Azure table
<input type="checkbox"/> AADNonInteractiveUserSignInLogs	Azure table
<input type="checkbox"/> AADServicePrincipalSignInLogs	Azure table

Choose a random table name, we will delete the table later:

1 Basics 2 Schema and transformation 3 Review

Table details

Start by adding a name and description for the table you're creating. On the next step, upload a sample of your custom log and adjust the table details to your needs.

Table name * ✓

Description

Data collection rule

Data collection rules (DCR) define the data coming into Azure Monitor and specify where that data should be sent or stored. [Learn more](#)

Data collection rule * ▾
[Create a new data collection rule](#)

Data collection endpoint * ▾

In the Schema and transformation part, you can upload your sample data again and create your transformation.

Basics Schema and transformation Review

Upload sample file Transformation editor

There was no timestamp field found in the sample provided. The transformation was updated to populate TimeGenerated column with the timestamp of data ingestion. Please use the transformation editor to review or update the logic of TimeGenerated column population in the destination table. [Learn more](#)

TimeGenerated	Key1	Key2	Key3	Key4
2023-05-20T07:42:35.739Z	String	2	true	key5=5 key6=6 key7=7

Once you have created your DCR and checked the template, you will see that the custom table is filled in as outputStream:

dc-api-tst-west-europe-001 | Export template

Data collection rule

Search Download Add to library Deploy Visualize template

Overview Activity log Access control (IAM) Tags Settings Locks Configuration Data sources Resources Automation Tasks (preview) Export template Support + troubleshooting New Support Request

To export related resources, select the resources from the Resource Group view then select the "Export template" option from the tool bar.

Include parameters

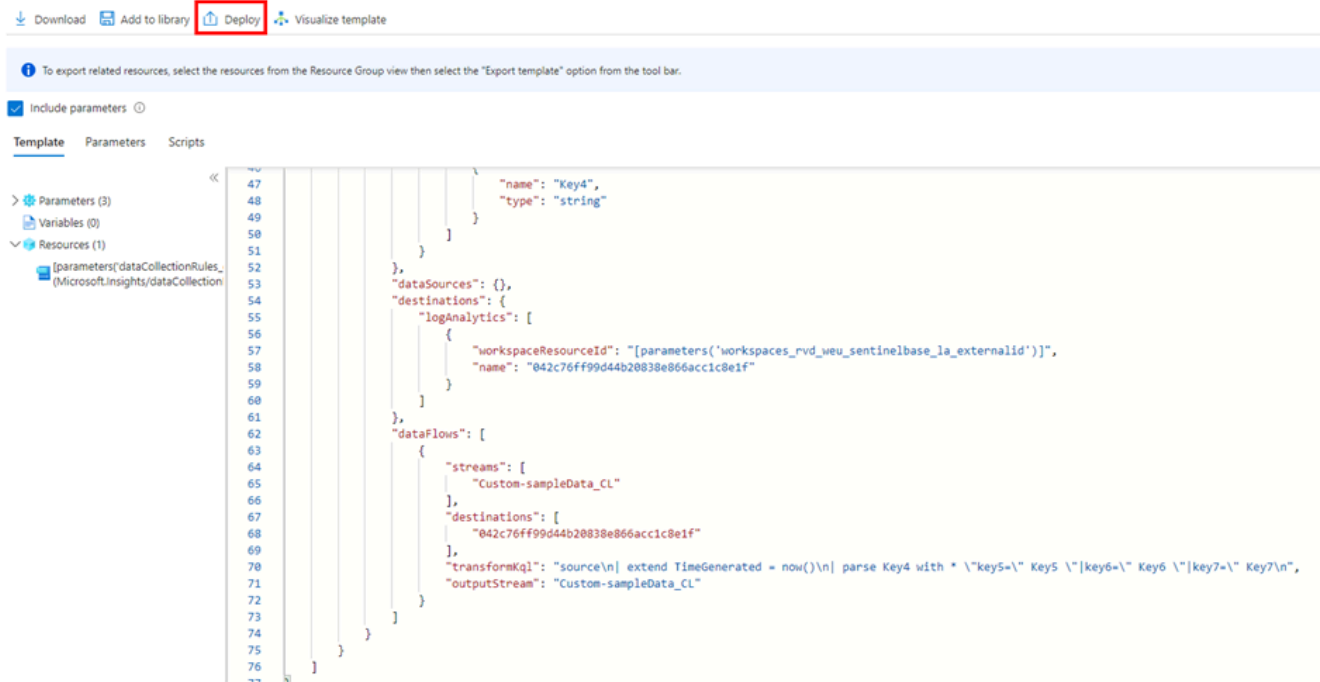
Template Parameters Scripts

```

47      "name": "Key4",
48      "type": "string"
49    }
50  },
51  },
52  "dataSources": {},
53  "destinations": {
54    "logAnalytics": [
55      {
56        "workspaceResourceId": "[parameters('workspaces_rvd_weu_sentinelbase_la_externalid')]",
57        "name": "042c76ff99d44b20838e066acc18e1f"
58      }
59    ]
60  },
61  "dataFlows": [
62    {
63      "streams": [
64        "Custom-sampleData_CL"
65      ],
66      "destinations": [
67        "042c76ff99d44b20838e066acc18e1f"
68      ],
69      "transformLogic": "[source() | extend TimeGenerated = now() | parse Key4 with * \"key5=\\\" Key5 \\\"|key6=\\\" Key6 \\\"|key7=\\\" Key7\\\"|n\",
70      \"outputStream\": \"Custom-sampleData_CL\"
71    }
72  ]
73  }
74  }
75  }
76  }

```

To change this to a supported standard table, click on Deploy:



Now edit the template:

Basics Review + create

Template



Custom template [↗](#)
1 resource



Edit template



Edit parameters



Visualize

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * [i](#)

RVDD

Resource group * [i](#)

RVD-WEU-SENTINELBASE-rg

[Create new](#)

Instance details

Region * [i](#)

(Europe) West Europe

Data Collection

Rules_dc_api_tst_westeurope_001_name

dc-api-tst-westeurope-001



Data Collection

Endpoints_RSECTVS01_externalid

Workspaces_rvd_weu_sentinelbase_la_exte

And change the outputStream to a standard Microsoft table (remember that standard tables need to have Microsoft- in front of them).

```
    },  
    "dataFlows": [  
      {  
        "streams": [  
          "Custom-sampleData_CL"  
        ],  
        "destinations": [  
          "042c76ff99d44b20838e866acc1c8e1f"  
        ],  
        "transformKql": "source\\n| extend TimeGenerated = now()\\n| parse Key4 with * \\\"key5=\\\" Key5 \\\"|key6=\\\" Key6 \\\"|key7=\\\" Key7\\n",  
        "outputStream": "Microsoft-SecurityEvent"  
      }  
    ]  
  }  
}
```

Once you have saved the template, you can deploy the template again to the same existing DCR with the newly created table. Remember that you will need to make sure that the **columns that are created or changed after the tranformationKql need to represent the columns of the standard table you chose in the outputStream**. After you have deployed the ARM template, you can delete the custom table we created for developing the transformation query.

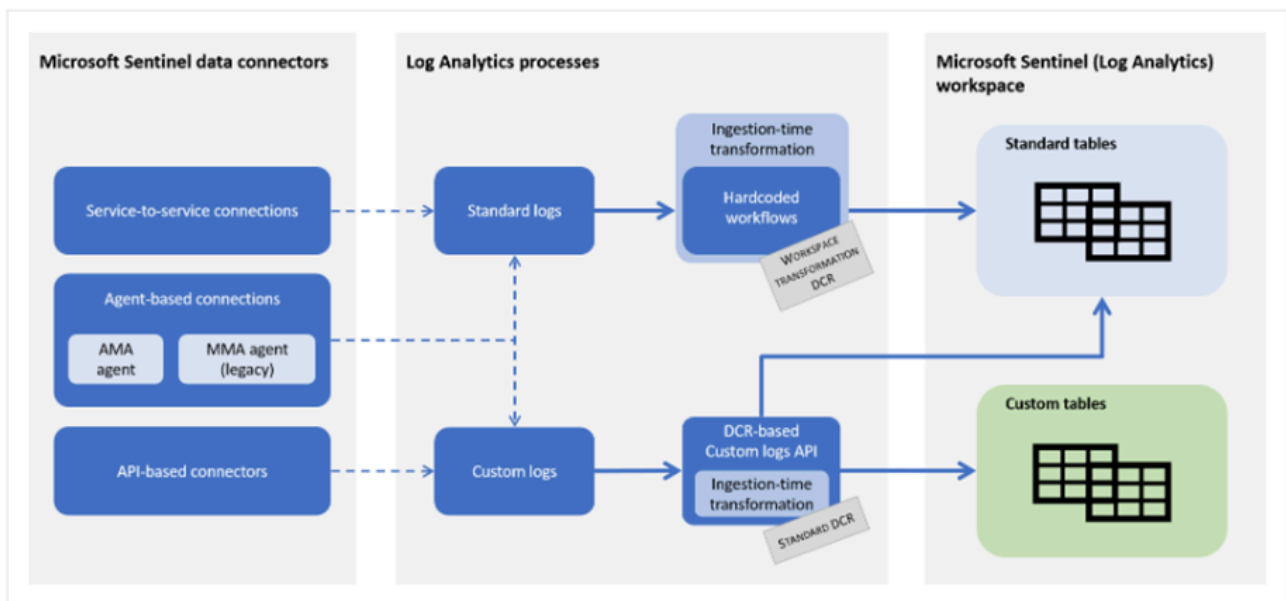
Why DCRs are so complicated

There are in my opinion a couple of problems with DCRs (at the time of writing this post) that make it so hard for people to understand how they work. First of all, the Microsoft documentation related to DCRs and transformation is all over the place. During the creation of this blog post, I had at a certain point 15 tabs open in my browser where I constantly had to switch between Microsoft documentation pages to gather all the info I needed.

Secondly, the two structures described by Microsoft of DCRs (being Custom Logs or Azure Monitoring Agent) are poorly documented and bring a lot of confusion. The only difference in essence is that Custom Logs

use streamDeclarations and needs a DCE, and that AMA uses data sources and doesn't need a DCE. What is also not clear in the documentation is that you can perfectly combine these two structures in one template (even though I wouldn't recommend it) and that outputStreams and transformationKql are perfectly useable in the AMA format (while it is only present in the Custom Logs format of the documentation).

The last problem I would like to describe is related to the below schema:



Don't get me wrong, the schema is good and quickly shows what DCRs are capable of. But one of the problems is that you can find on the schema that standard DCRs can send data to standard and custom tables. However, in some DCR creation flows in the portal you only have the option to choose a custom table (referring to the creating standard DCR for Log Ingestion API part), which brings again a lot of confusion. Another example is that you can create transformations for DCRs that use the Custom Logs format while this is not possible for DCRs that use the AMA format, even though you can perfectly add a transformationKql to an AMA formatted DCR and redeploy the template.

Even though DCRs are in my opinion complicated, I think they are very powerful and are a must-learn and understand for anyone who is regularly deploying advanced data connectors in Microsoft Sentinel.

Logstash and DCRs

In this blog post, we focussed on how DCRs are created, how they work, and what the pitfalls are. Even though we will not be talking about Logstash in this post, I would like to mention that Logstash is a very powerful alternative to transformations if you want to do data normalization and filtering before it is being sent to Sentinel. It even doesn't have to be an alternative to transformations, since you can now [combine Logstash and transformations by using the new DCR-supported output plugin of Logstash](#). I highly recommend checking Logstash and the DCR support out, since it can be a very powerful combination if you want to create custom on-premise data connectors and still want to ingest them in standard log analytics tables. Logstash uses the Log Ingestion API, which means you can use standard DCRs in the custom logs format to manipulate your data.

If you want to learn more about using Logstash and DCRs, I [recommend reading the post created by Koos Goossens](#)! Once you understand this setup, make sure to [harden your Logstash connections to Microsoft Sentinel](#).

Golden tip

I recently stumbled on a very cool workbook created by the community, which **helps you with creating and visualizing DCRs in your workspace**. An explanation of the workbook and how it works can be found here: <https://techcommunity.microsoft.com/t5/microsoft-sentinel->

[blog/create-edit-and-monitor-data-collection-rules-with-the-data-ba-p/3810987](https://hybridbrothers.com/demystifying-data-collection-rules-and-transformations/blog/create-edit-and-monitor-data-collection-rules-with-the-data-ba-p/3810987).

One of the most powerful tools in this workbook is the feature where you can create DCRs for Windows servers based on NSA and MITRE categories:

Create New DCR Rule

[New Linux DCR](#)
[New Windows DCR](#)
[New Table Transformation DCR](#)
[New Custom Log](#)
[Essentials](#)
[Reset](#)

[NSA Category Events](#)
[MITRE Category Events](#)
[Recommended Event ID](#)
[AMA Data Connector](#)
[File Path DCR](#)

Ingestion Tier: ☐ None ☐ Minimal ☒ Common

Antivirus ☒ Ingest ☐ Ignore
 Privilege Escalation ☒ Ingest ☐ Ignore
 Lateral Movement ☒ Ingest ☐ Ignore

Initial Access ☒ Ingest ☐ Ignore
 Defense Evasion ☒ Ingest ☐ Ignore
 Collection ☒ Ingest ☐ Ignore

Execution ☒ Ingest ☐ Ignore
 Credential Access ☒ Ingest ☐ Ignore
 Command and Control ☒ Ingest ☐ Ignore

Persistence ☒ Ingest ☐ Ignore
 Discovery ☒ Ingest ☐ Ignore
 Impact ☒ Ingest ☐ Ignore

Manual Events: Excluded Events:

Manually Entered IDs:

EventIDs:

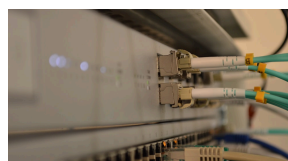
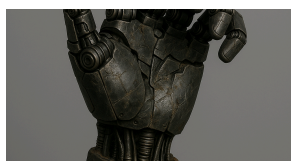
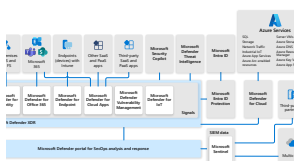
Excluded IDs:

EventIDs:

Full List of Manual and MITRE IDs:

EventIDs [1, 4, 10, 11, 46, 299, 300, 321, 324, 325, 327, 340, 354, 403, 404, 410, 411, 412, 413, 431, 500, 501, 699, 800, 808, 984, 1100, 1102, 1107, 1108, 1116, 1151, 4103, 4104, 4608, 4610, 4611, 4614, 4622, 4624, 4625, 4634, 4647, 4648, 4649, 4656, 4657, 4661, 4662, 4663, 4665, 4666, 4667, 4670, 4672, 4673, 4674, 4675, 4688, 4689, 4697, 4698, 4699, 4700, 4702, 4704, 4705, 4716, 4717, 4718, 4719, 4720, 4722, 4723, 4724, 4725, 4726, 4727, 4728, 4729, 4732, 4733, 4735, 4737, 4738, 4739, 4740, 4742, 4744, 4745, 4746, 4750, 4751, 4752, 4754, 4755, 4756, 4757, 4760, 4761, 4762, 4764, 4767, 4768, 4769, 4771, 4774, 4778, 4779, 4781, 4793, 4794, 4797, 4798, 4799, 4800, 4801, 4802, 4803, 4825, 4826, 4870, 4886, 4887, 4888, 4889, 4898, 4902, 4904, 4905, 4907, 4931, 4932, 4933, 4946, 4948, 4956, 4985, 5024, 5033, 5059, 5136, 5137, 5140, 5142, 5145, 5632, 6144, 6145, 6272, 6416, 7000, 7009, 7045, 33200]

Which is in my opinion far more powerful than ingesting Event IDs via the ‘all, common, minimal’ filter. Or having to plot Event IDs to the MITRE framework manually. Like I did in my previous post.



**Transition
from**

**Detecting
non-**

**MDE
Device**

Microsoft Sentinel to Defender XDR - Practical challenges

Introduction
Microsoft announced on the...

Jul 4, 2025 12 min read

privileged Windows Hello abuse

Introduction I recently followed a live session of Dirk...

Apr 26, 2025 16 min read

Discovery - Improving the monitored network page

Introduction This blogpost is probably the first ...

Mar 19, 2025 6 min read

Hybrid Brothers © 2025

[Sign up](#) [Privacy policy](#)

Powered by Ghost